IFIP AICT 471

Jaap-Henk Hoepman
Stefan Katzenbeisser
(Eds.)

ICT Systems Security and Privacy Protection

31st IFIP TC 11 International Conference, SEC 2016
Ghent, Belgium, May 30 – June 1, 2016
Proceedings

Springer

IFIP Advances in Information and Communication Technology 471

Editor-in-Chief

Kai Rannenberg, Goethe University Frankfurt, Germany

Editorial Board

IFIP – The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the first World Computer Congress held in Paris the previous year. A federation for societies working in information processing, IFIP's aim is two-fold: to support information processing in the countries of its members and to encourage technology transfer to developing nations. As its mission statement clearly states:

> *IFIP is the global non-profit federation of societies of ICT professionals that aims at achieving a worldwide professional and socially responsible development and application of information and communication technologies.*

IFIP is a non-profit-making organization, run almost solely by 2500 volunteers. It operates through a number of technical committees and working groups, which organize events and publications. IFIP's events range from large international open conferences to working conferences and local seminars.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is generally smaller and occasionally by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is also rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

IFIP distinguishes three types of institutional membership: Country Representative Members, Members at Large, and Associate Members. The type of organization that can apply for membership is a wide variety and includes national or international societies of individual computer scientists/ICT professionals, associations or federations of such societies, government institutions/government related organizations, national or international research institutes or consortia, universities, academies of sciences, companies, national or international associations or federations of companies.

More information about this series at http://www.springer.com/series/6102

Jaap-Henk Hoepman · Stefan Katzenbeisser (Eds.)

ICT Systems Security and Privacy Protection

31st IFIP TC 11 International Conference, SEC 2016
Ghent, Belgium, May 30 – June 1, 2016
Proceedings

Springer

Editors
Jaap-Henk Hoepman
Institute for Computing and Information Sciences
Radboud University Nijmegen
Nijmegen
The Netherlands

Stefan Katzenbeisser
Security Engineering Group
Technische Universität Darmstadt
Darmstadt
Germany

ISSN 1868-4238 ISSN 1868-422X (electronic)
IFIP Advances in Information and Communication Technology
ISBN 978-3-319-33629-9 ISBN 978-3-319-33630-5 (eBook)
DOI 10.1007/978-3-319-33630-5

Library of Congress Control Number: 2016937373

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Preface

It is our great pleasure to present the proceedings of the 31st IFIP International Conference on ICT Systems Security and Privacy Protection, which was held in Ghent, Belgium, between May 30 and June 1, 2016. IFIP SEC conferences are the flagship events of the International Federation for Information Processing (IFIP) Technical Committee 11 on Information Security and Privacy Protection in Information Processing Systems (TC-11).

Continuing the tradition of previous years, we sought for a balanced program that covers all significant aspects of information security, ranging from software security over platform security to human factors. The selection of papers was a highly challenging task. We received 145 submissions in response to our call for papers, of which six were withdrawn before the reviewing process started. From these 139 submissions we selected 27 full papers to be presented at the conference, based on their significance, novelty, and technical quality. Each paper received at least three reviews by members of the Program Committee.

We wish to thank all contributors who helped make IFIP SEC 2016 a success: the authors who submitted their latest research results to the conference, as well as the members of the Program Committee who devoted significant amounts of their time to evaluate all submissions. We would like in particular to thank the organizing chair Vincent Naessens and the general chair Bart de Decker for their support and their efforts to organize the conference in beautiful Ghent.

We hope that this proceedings volume provides inspirations for future research in the area of information security!

March 2016

Stefan Katzenbeisser
Jaap-Henk Hoepman

Organization

Program Committee

Gunes Acar	KU Leuven, Belgium
Luca Allodi	University of Trento, Italy
Frederik Armknecht	Universität Mannheim, Germany
Vijay Atluri	Rutgers University, USA
Matt Bishop	University of California at Davis, USA
Rainer Boehme	University of Innsbruck, Austria
Joan Borrell	Universitat Autònoma de Barcelona, Spain
Joppe Bos	NXP Semiconductors, Leuven, Belgium
Dagmar Brechlerova	Euromise Prague, Czech Republic
Christina Brzuska	Hamburg University of Technology, Germany
William Caelli	IISEC Pty Ltd., Australia
Jan Camenisch	IBM Research — Zurich, Switzerland
Iliano Cervesato	Carnegie Mellon University, USA
Eric Chan-Tin	Oklahoma State University, USA
Nathan Clarke	Centre for Security, Communication and Network Research, University of Plymouth, UK
Frédéric Cuppens	Telecom Bretagne, France
Nora Cuppens-Boulahia	Telecom Bretagne, France
Ernesto Damiani	University of Milan, Italy
Sabrina De Capitani di Vimercati	University of Milan, Italy
Mourad Debbabi	Concordia University, Canada
Andreas Dewald	University of Mannheim, Germany
Gurpreet Dhillon	Virginia Commonwealth University, USA
Theo Dimitrakos	Security Research Centre, BT Group CTO, UK
Jana Dittmann	University of Magdeburg, Germany
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Spain
Paul Dowland	Plymouth University, UK
Hannes Federrath	University of Hamburg, Germany
Simone Fischer-Hübner	Karlstad University, Sweden
William Michael Fitzgerald	United Technologies Research Centre Ireland, Ltd., Ireland
Sara Foresti	University of Milan, Italy
Felix Freiling	Friedrich-Alexander-Universität, Germany
Lothar Fritsch	Norsk Regnesentral — Norwegian Computing Center, Norway
Steven Furnell	Plymouth University, UK

Lynn Futcher	IFIP WG 11.8 (Vice-chair), South Africa
Dieter Gollmann	Hamburg University of Technology, Germany
Stefanos Gritzalis	University of the Aegean, Greece
Seda Gürses	NYU, USA
Marit Hansen	Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, Germany
Karin Hedström	Swedish Buisiness School, Örebro University, Sweden
Andreas Heinemann	Hochschule Darmstadt — University of Applied Sciences, Germany
Dominik Herrmann	University of Hamburg, Germany
Michael Herrmann	KU Leuven ESAT/COSIC, iMinds, Belgium
Alejandro Hevia	University of Chile, Chile
Jaap-Henk Hoepman	Radboud University Nijmegen, The Netherlands
Ralph Holz	NICTA, Australia
Xinyi Huang	Fujian Normal University, China
Sushil Jajodia	George Mason University, USA
Lech Janczewski	The University of Auckland, New Zealand
Christian Damsgaard Jensen	Technical University of Denmark
Thomas Jensen	Inria, France
Martin Johns	SAP Research, Germany
Wouter Joosen	Katholieke Universiteit Leuven, Belgium
Audun Josang	University of Oslo, Norway
Sokratis Katsikas	University of Piraeus, Greece
Stefan Katzenbeisser	TU Darmstadt, Germany
Florian Kerschbaum	SAP, Germany
Dogan Kesdogan	Universität Regensburg, Germany
Kwangjo Kim	KAIST, South Korea
Valentin Kisimov	UNWE, Bulgaria
Zbigniew Kotulski	Warsaw University of Technology, Poland
Stefan Köpsell	TU Dresden, Germany
Ronald Leenes	Tilburg University – TILT, The Netherlands
Luigi Logrippo	Université du Québec en Outaouais, Canada
Javier Lopez	University of Malaga, Spain
Emil Lupu	Imperial College, UK
Stephen Marsh	UOIT, UK
Fabio Martinelli	IIT-CNR, Italy
Michael Meier	University of Bonn, Fraunhofer FKIE, Germany
Martin Mulazzani	SBA Research, Austria
Yuko Murayama	Iwate Prefectural University, Japan
Eiji Okamoto	University of Tsukuba, Japan
Daniel Olejar	Comenius University, Slovakia
Federica Paci	University of Southampton, UK
Jakob Illeborg Pagter	Centre for IT Security, The Alexandra Institute Ltd., Denmark
Sebastian Pape	Goethe University Frankfurt, Germany
Philippos Peleties	Cyprus Computer Society, Cyprus

Günther Pernul	Universität Regensburg, Germany
Andreas Peter	University of Twente, The Netherlands
Gilbert Peterson	US Air Force Institute of Technology, USA
Wolter Pieters	TBM-ESS, Delft University of Technology, The Netherlands
Joachim Posegga	University of Passau, Germany
Sihan Qing	Peking University, China
Kai Rannenberg	Goethe University Frankfurt, Germany
Indrajit Ray	Colorado State University, USA
Indrakshi Ray	Colorado State University, USA
Konrad Rieck	University of Göttingen, Germany
Carlos Rieder	ISEC AG, Switzerland
Yves Roudier	EURECOM, France
Mark Ryan	University of Birmingham, UK
Peter Ryan	University of Luxembourg, Luxembourg
Pierangela Samarati	Università degli Studi di Milano, Italy
Thierry Sans	Carnegie Mellon University in Qatar
Damien Sauveron	University of Limoges, France
Ingrid Schaumüller-Bichl	Upper Austrian University of Applied Sciences Campus Hagenberg, Austria
Björn Scheuermann	Humboldt University of Berlin, Germany
Sebastian Schinzel	Münster University of Applied Sciences, Germany
Joerg Schwenk	Ruhr-Universität Bochum, Germany
Anne Karen Seip	Finanstilsynet, Norway
Jetzabel Maritza Serna Olvera	Universidad Politecnica de Cataluña, Spain
Abbas Shahim	VU University Amsterdam, The Netherlands
Haya Shulman	Technische Universität Darmstadt, Germany
Adesina S. Sodiya	Federal University of Agric, Abeokuta, Nigeria
Radu State	University of Luxembourg, Luxembourg
Jakub Szefer	Yale University, USA
Kerry-Lynn Thomson	Nelson Mandela Metropolitan University, South Africa
Nils Ole Tippenhauer	Singapore University of Technology and Design, Singapore
Carmela Troncoso	Gradiant, Spain
Markus Tschersich	Goethe University Frankfurt, Germany
Pedro Veiga	University of Lisbon, Portugal
Michael Vielhaber	Hochschule Bremerhaven, Germany
Melanie Volkamer	Technische Universität Darmstadt, Germany
Rossouw Von Solms	Nelson Mandela Metropolitan University, South Africa
Jozef Vyskoc	VaF, Slovak Republic
Lingyu Wang	Concordia University, Canada
Christian Weber	Ostfalia University of Applied Sciences, Germany
Edgar Weippl	SBA Research, Austria
Tatjana Welzer	University of Maribor, Slovenia
Steffen Wendzel	Fraunhofer FKIE, Germany

Gunnar Wenngren	AB Wenngrens i Linköping, Sweden
Jeff Yan	Newcastle University, UK
Zhenxin Zhan	University of Texas at San Antonio, USA
André Zúquete	IEETA, University of Aveiro, Portugal

Contents

Phishing and Data Sharing

Social Networks

Software Vulnerabilities

TPM and Internet of Things

Sidechannel Analysis

Software Security

Privacy

Cryptographic Protocols

Coercion-Resistant Proxy Voting

Oksana Kulyk[1(✉)], Stephan Neumann[1], Karola Marky[1], Jurlind Budurushi[1], and Melanie Volkamer[1,2]

[1] Technische Universität Darmstadt/CASED, Darmstadt, Germany
{oksana.kulyk,stephan.neumann,karola.marky,jurlind.budurushi,
melanie.volkamer}@secuso.org
[2] Karlstad University, Karlstad, Sweden

Abstract. In general, most elections follow the principle of equality, or as it came to be known, the principle of "one man – one vote". However, this principle might pose difficulties for voters, who are not well informed regarding the particular matter that is voted on. In order to address this issue, a new form of voting has been proposed, namely *proxy voting*. In proxy voting, each voter has the possibility to delegate her voting right to another voter, so called *proxy*, that she considers a trusted expert on the matter. In this paper we propose an end-to-end verifiable Internet voting scheme, which to the best of our knowledge is the first scheme to address voter coercion in the proxy voting setting.

1 Introduction

Democratic elections represent an important value in many modern states. This is not limited to governments, also companies and other organizations are constituted in democratic elections. In general, most democratic elections follow the principle of equal elections, meaning that one person's vote should be worth as much as another's, i.e. one man – one vote [14]. However, this principle often represents a challenge to voters, who are not sufficiently informed regarding the particular matter that is voted on. In order to address this issue, a new form of voting has been proposed, the so called *proxy voting*. In proxy voting, each eligible voter has the possibility to delegate her voting right to another eligible voter, so called *proxy*, that she considers a trusted expert on the matter.

There already exist few proxy voting implementations, provided by different organizations. Two widely known implementations are LiquidFeedback[1] and Adhocracy[2]. Further proxy voting proposals are the approaches proposed in [19,21].

However, all existing proxy voting proposals fail to address the issue of voter coercion: namely, the case when the adversary threatens the voter to vote in a particular way, or to abstain from voting. This issue has been commonly considered for non-proxy Internet voting, and a number of Internet voting schemes

[1] http://liquidfeedback.org/, last accessed January, 7, 2016.
[2] https://adhocracy.de/, last accessed January, 7, 2016.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 3–16, 2016.
DOI: 10.1007/978-3-319-33630-5_1

have been proposed, that address the problem of coercion, e.g. by providing coercion resistance [12] or coercion evidence [8]. In this paper, we build upon [6,12] and an extension proposed by Spycher *et al.* [18] to propose a coercion resistant end-to-end verifiable Internet proxy voting scheme.

This paper is structured as follows: In Sect. 2 we identify and derive security requirements that are relevant for proxy voting. Section 3 introduces the fundamentals used for our proposal, which we present in Sect. 4. In Sect. 5 we evaluate the security of our proposal with respect to the requirements. Section 6 summarizes our contributions and provides directions for future research.

2 Requirements for Proxy Voting

The following **functional requirements** should be provided by a proxy voting system:

Delegation Cancellation. If the voter for any reasons decides to vote herself, she should be able to cancel the delegation at any point of time before the tallying.

Delegation Back-Up. The voter can assign up to T priorities to her proxies. Only the vote from the proxy having highest priority will be included in the vote count. This functionality is useful if the voter wants to have a "back-up" delegation, in case her first choice of a proxy abstains from the election.

The **security requirements** in Internet voting have been thoroughly investigated in the literature, and both formal [7] and informal [13,16] definitions have been proposed. In this work, we aim to address the following security requirements for the election.

Vote Integrity. All votes cast by eligible voters should be included in the result.

Availability. It should be possible to compute election result even if some of the involved entities are faulty.

Vote Secrecy for Voters. The adversary should not be able to learn how a voter has voted.

We aim at achieving the following security requirements for the delegation process:

Delegation Integrity. Only the proxy having a valid permit from the voter should be able to cast a vote on this voter's behalf. The proxy should not be able to alter the priority given to them by the voter.

Delegation Availability. A proxy should not be selectively prevented from having the votes delegated to her.

Vote Secrecy for Proxies. The adversary should not be able to learn how a proxy has voted.

Delegation Privacy. There should not be a link between a voter's identity and her selected proxy. Furthermore, it should not be possible to reveal whether a voter has delegated a vote or cast it herself.

Delegation Unprovability. The proxy should not be able to prove to anyone how many votes have been delegated to her. Moreover, the proxy can not gain any knowledge about the actual number the incoming delegations.

Note, that as we want to ensure coercion resistance, we require that vote secrecy for both voters and proxies should be ensured also for the case when the adversary is capable of communicating with the voter or proxy.

3 Background

In this section we introduce the fundamentals used to design our coercion-resistant verifiable proxy voting scheme.

3.1 Cryptographic Primitives

In the following we describe the cryptographic primitives our scheme relies on. Hereby, $\mathbb{G}_q$ denotes a cyclic multiplicative group with order q and $\mathbb{Z}_q$ denotes the cyclic additive group of integers modulo q.

Zero-Knowledge Proofs. In order to prove the correctness of statements within the voting scheme without revealing anything beyond the correctness *zero-knowledge proofs* are employed. For this sake, techniques such as proving the knowledge of discrete logarithm [17] or discrete logarithm equality [5] are being used. These proofs can be made non-interactive by employing the strong version of the Fiat-Shamir heuristic suggested in [3]. An important extension of such proofs are *designated-verifier proofs* described in [11]. Given the verifier's public key, these proofs convince only the designated verifier about the correctness, rather than the general public.

Linear Encryption. In some parts of our scheme, we use a modified encryption scheme suggested in [4] (further denoted as *LE-ElGamal*). This scheme is semantically secure under the DLIN assumption which is implied in groups where the DDH assumption holds. Namely, let $pk = (g_1, g_2, h) \in \mathbb{G}_q^3$ be the public keys of the encryption and $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ the private keys with $g_1^{x_1} = g_2^{x_2} = h$. If the keys are jointly generated by multiple parties with x_1, x_2 as threshold-shared secrets, then, according to [2] at least 2/3 of the parties have to be honest.

The message $m \in \mathbb{G}_q$ is encrypted as follows: two random values $(r_1, r_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ are chosen and then the encryption - denoted as $\{\{m\}\}_{pk} \in \mathbb{G}_q^3$ - is calculated as $(c_1, c_2, c_3) = (g_1^{r_1}, g_2^{r_2}, m \cdot h^{r_1+r_2})$. The decryption then proceeds

as $m = c_3 \cdot c_1^{-x_1} \cdot c_2^{-x_2}$. Ciphertexts can then be reencrypted by multiplying a ciphertext by an encryption of 1 using a random value $(r'_1, r'_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$. Further operations used together with the ElGamal encryption, such as mix net shuffle, well-formedness proofs and plaintext equivalence tests can be adjusted for LE-ElGamal as well.

Plaintext Equivalence Tests. In order to check whether a given ciphertext c encrypts the same plaintext as another ciphertext c' without revealing any additional information about plaintexts, *plaintext-equivalence tests* [10] can be employed. This can be performed by the holders of an encryption secret key and consists of jointly decrypting the value $(c/c')^r$ given a secretly shared random value $r \in \mathbb{Z}_q$ which results in 1 in case the plaintexts are equal or in random value otherwise.

Proxy Reencryption. Let $\{m\}_{pk_1}$ be a ciphertext encrypting message m with ElGamal public key $pk_1 = (g_1, h_1)$. Given the knowledge of corresponding secret key $x_1 = \log_{g_1} h_1$ which can also be a shared secret between several participants the method described in [9] allows for computing a new ciphertext $\{m\}_{pk_2}$, that encrypts the same message using a different ElGamal public key pk_2.

Mix Nets. Important components in electronic voting systems are *mix net schemes* which are used for anonymizing lists of ciphertexts $e_1, ..., e_N$. In addition to ensure the integrity of the shuffle a number of zero-knowledge proofs have been proposed in the literature. The most efficient schemes - up to now - are presented in [1,20]. A modification of such proofs can be used to mix tuples of ciphertexts $(\bar{e_1}, ..., \bar{e_N})$ with $\bar{e_i} = (e_{i,1}, ..., e_{i,k})$ while preserving the ordering within the tuple.

3.2 JCJ/Civitas Scheme

For our goal to provide a scheme for coercion resistant proxy voting we chose the JCJ/Civitas voting scheme proposed in [12] and then extended and implemented in [6] as basis. The coercion resistance of the scheme is based on the application of voting credentials. These credentials are used to authorize votes from eligible voters. In case a coercer demands the credential from a voter she can simply provide a fake credential which could not be distinguished from the real one by the coercer. We briefly outline the scheme below.

Setup and Registration. Prior to the election the election supervisor announces the different election authorities, namely the *registrar*, *registration tellers* and *tabulation tellers* and publishes their respective public keys on the bulletin board. The registrar publishes the electoral register which contains the identity, the public registration key, and the public designation key of each voter. Building upon a homomorphic cryptosystem the tabulation tellers generate an election key pair in a distributed manner and publish the respective public key pk on the bulletin board.

For each voter V_i each registration teller $j = 1, ..., N$ generates a credential share $c_{i,j}$ and publishes its encryption next to the respective voter's identity on the bulletin board from which the encryption of the voting credential $E_i = \{c_i\}_{pk} = \prod_{j=1}^{N} \{c_{i,j}\}_{pk}$ can be computed by multiplying all the individual credential shares. The shares $c_{i,j}$ in plaintext together with corresponding designated-verifier correctness proofs are then being forwarded to the voter. Now the voter can use them to compute their secret voting credential $c_i = \prod_{i=1}^{N} c_{i,j}$. Finally, the encrypted credentials E_i are being shuffled via mix net.

Voting. The voters use anonymous channels for vote casting. As her vote, the voter casts a tuple

$$\langle A_i = \{o\}_{pk}, C_i = \{c_i\}_{pk}, \sigma \rangle$$

with o as a chosen voting option and σ as well-formedness proof for A_i and proof of plaintext knowledge for C_i. The tuple is sent to one of the available *ballot boxes* which stores the votes. In case the voter is forced to reveal her credential to a coercer she can give a fake credential c' instead while the coercer is not able to distinguish it from a real one.

Tallying. The votes with invalid proofs are excluded and the plaintext-equivalence tests are used for identifying the votes with duplicated credentials which are handled according to the rules concerning vote updating. The remaining tuples $\langle A_i, C_i \rangle$ are being shuffled and of votes are being anonymized with mix net shuffling. Afterwards, plaintext-equivalence tests are applied for checking the validity of the voting credential by each C_i with each authorized credential from the shuffled list E_i. For the votes with valid credentials the voting options A_i are being decrypted.

Security assumptions that JCJ/Civitas relies on:

1. The adversary is not capable of simulating the voters during the registration process. The adversary is also not present the registration phase.
2. At least one registration teller is trusted and the communication channel between this teller and the voter is untappable[3].
3. The voting client is trusted.
4. At least k out of n tabulation tellers do not fail during decryption.
5. At least $n - k + 1$ out of n tabulation tellers are trusted not to reveal their key shares.
6. The channels between voters and voting system used for vote casting are anonymous.
7. The DDH assumption and the RSA assumption hold an a random oracle is implemented via cryptographic hash function.
8. At least one ballot box to which the cast votes are submitted is correct.

In addition to the security requirements, it is assumed that the voters are capable of handling the credentials, e.g. by using some kind of secure credential management.

[3] That is, the adversary is incapable of reading the messages sent over the channel.

4 Proposed Proxy Voting Scheme

To tailor our JCJ/Civitas extension towards proxy voting we introduce a new kind of credentials, so called *delegation credentials*. In addition to a unique voter credential in JCJ/Civitas, each voter i obtains a list of prioritized *delegation credentials*. To delegate a vote with a certain priority j the voter selects the j-th credential from her list and forwards it to the intended proxy. Voters are allowed to forward different credentials with different priorities to different proxies. Throughout the tallying phase for each voter only the vote cast with the highest priority is counted. Due to the fact that delegation credentials are generated on the same principles as the voting credentials in the original scheme the security of our extension also relies on the fact that the delegating credentials can be faked by the voter in case of coercion.

4.1 Necessary Modifications

We describe the modifications to the JCJ/Civitas scheme that are needed for implementing the delegation while ensuring the requirements listed in Sect. 2.

Ballot Clustering. Within the JCJ/Civitas scheme coercion-resistance is achieved by breaking the link between a voter's identity and votes cast in her name, both real and fake votes. The introduction of prioritized delegation credentials requires a relation between different credentials being maintained throughout the vote tallying phase. Retaining such a relation might however cause vulnerabilities with regard to coercion. To address these challenges we build upon proposals from scientific literature. Spycher *et al.* [18] present a JCJ extension towards linear tallying time. Therefore, the authors propose to assign identifiers to cast (real and fake) votes. During vote tallying after anonymization cast votes are only compared against the public credential assigned to their respective identifier. This reduces the tallying complexity from quadratic to linear regarding the number of cast votes. We build upon this approach: votes cast by the voter or delegated to different proxies share the same identifier such that within the set of votes sharing the same identifier the vote with the highest priority is tallied.

Delegation Server. Forwarding voting credentials to proxies results in a coercion vulnerability: The adversary might coerce a proxy to forward all received voting credentials. In order to test whether the proxy complies the adversary could anonymously delegate a credential to her and check whether this credential is being forwarded back to her. We address this problem by introducing a new entity - possibly implemented in a distributed way - that functions as *delegation server* (DS). The underlying idea is that proxies do not receive delegated credentials directly from the voter. Instead the voter blinds her credential and sends it to the DS (over an anonymous and untappable channel) which forwards an anonymization of the blinded credential to the proxy. The unblinding value is sent to the proxy over the private and anonymous channel.

Inclusion of Linear Encryption. To prevent unauthorized usage of voting credentials the JCJ/Civitas scheme forces the voter to include the plaintext knowledge proof for the ciphertext encrypting the credential. This solution, however, is inapplicable to the delegation process since the proxy does not get to know the value of the credential as outlined above. We address this challenge by publishing voting credentials (in the registration and voting phases) encrypted with linear encryption rather than ElGamal encryption. On the other hand for delegating her vote the voter encrypts their delegating credential with standard ElGamal using (g_2, h) as an encryption key. The resulting tuple $\{c\}_{pk} = (a = g_2^r, b = ch^r)$ together with other necessary information (see Sect. 3.2) is being forwarded to the proxy. If the proxy wants to cast the vote she chooses a random value of s and computes (g_1^s, a, bh^s) which is an LE-ElGamal encryption of c with randomness values r, s for which the proxy also can prove the knowledge of s as $\log_{g_1} g_1^s$.

4.2 Scheme Description

In this section we provide a detailed description of our proposed scheme.

Preliminary Stage. During this stage the keys used for encrypting votes and/or credentials are generated. The DS generates ElGamal keys with $pk_D = (g_D, h_D)$ as public key. The tabulation tellers distributively generate LE-ElGamal keys $pk_T = (g_1, g_2, h_T)$. We further denote $\{m\}_{pk_T}$ as ElGamal encryption with (g_2, h_T) as corresponding key and $\{\{m\}\}_{pk_T}$ as LE-ElGamal encryption.

The list of proxies $D_1, ..., D_d$ is being made public together with their public keys used for signing and designated-verifier proofs. For the sake of simplicity we assume that each proxy is eligible to vote herself as well. Furthermore, anonymous channels that enable communication between the proxies and the delegation servers as well as between proxies and the rest of the voters are established.

Setup Phase. Opposed to the standard JCJ/Civitas scheme, each registration teller generates T credential shares at random for each voter. Analogously to the JCJ/Civitas scheme, the encrypted credentials are publicly assigned to the respective voters whereby the order of the credential shares is of central importance to the delegation process. A public identifier, e.g. the position of the respective voter in the electoral roll is assigned to each voter. After the setup phase the bulletin board contains T credentials for every voter $V_1, ..., V_k$ (see Table 1) as well as individual credential shares from each of N registration tellers for each priority $1, ..., T$. We consider the lower number to denote the higher priority.

Registration. The registration phase remains identical to the standard JCJ/Civitas scheme except the fact that each registration teller releases T ordered credential shares to the voter. The voter can then verify whether the received shares from the tellers for a credential $c_i^{(j)}$ correspond to the encrypted shares published on the bulletin board near id_i and priority j.

Table 1. Content of the bulletin board after the setup phase of the extended scheme.

ID	Priority	Credential shares
id_1	1	$\{\{c_1^{1,1}\}\}_{pk_T}, \ldots, \{\{c_1^{1,N}\}\}_{pk_T}$
$\vdots$	$\vdots$	$\vdots$
id_k	T	$\{\{c_n^{T,1}\}\}_{pk_T}, \ldots, \{\{c_n^{T,N}\}\}_{pk_T}$

Before the voting, the voter merges her $N \cdot T$ credential shares as follows:

$$\begin{aligned} c_i^{(1)} &= c_i^{1,1} \cdot c_i^{1,2} \cdot \cdots \cdot c_i^{1,N} \\ &\vdots \quad \vdots \\ c_i^{(T)} &= c_i^{T,1} \cdot c_i^{T,2} \cdot \cdots \cdot c_i^{T,N} \end{aligned}$$

Voting. To cast a vote (without considering delegation) for voting option o voter i prepares the following tuple:

$$\langle \{\{id_i\}\}_{pk_T}, \{\{c_i^{(j)}\}\}_{pk_T}, \{\{o\}\}_{pk_T}, \sigma \rangle$$

Here σ signifies both the well-formedness proofs for o as well as proof of randomness knowledge for $\{\{c_i^{(j)}\}\}_{pk_T}$: namely, given $\{\{c_i^{(j)}\}\}_{pk_T} = (c_1, c_2, c_3) = (g_1^{r_1}, g_2^{r_2}, c_i^{(j)} h^{r_1+r_2})$ the voter proves the knowledge of randomness r_1 as $\log_{g_1} c_1$. The value of j is chosen depending on the voter's delegations where we distinguish the following cases:

1. If the voter does not intend to delegate her vote at a later point in time she sets $j = 1$.
2. If the voter might intend to delegate her vote at a later point in time she sets j as the lowest available priority: that is $j = T$ in case she did not delegate any vote yet or $j = j_d - 1$ if j_d is the highest priority that was already delegated.

Additionally, the voter i casts her identifier id_i in an encrypted manner which later serves for clustering ballots from the same voter with different credentials.

Delegating. To delegate her vote with priority $j = 2, \ldots, T$ to the proxy D_k the following protocol (see Fig. 1) is executed.

1. The voter i chooses a random value x and sends the following tuple to one or more of the DS:
$$\langle \{\{id_i\}\}_{pk_T}, \{(c_i^{(j)})^x\}_{pk_D}, \sigma, id_{D_k} \rangle$$
Here $c_i^{(j)}$ is the j-th credential from her credential list $(c_i^{(1)}, \ldots, c_i^{(T)})$, id_i is the voter's index, σ is the proof of plaintext knowledge for $\{(c_i^{(j)})^x\}_{pk_D}$ and id_{D_k} is the identifier, e.g. the public key, of the chosen proxy. The voter also sends $x, \{\{id_i\}\}_{pk_T}$ to D_k over a private channel.

2. The DS computes $\{(c_i^{(j)})^x\}_{pk_T}$ from $\{(c_i^{(j)})^x\}_{pk_D}$ using proxy reencryption scheme and a designated-verifier proof using the public designated-verifier key of D_k that both $\{(c_i^{(j)})^x\}_{pk_T}$ and $\{(c_i^{(j)})^x\}_{pk_D}$ encrypt the same plaintext. The proof and the values of $\{(c_i^{(j)})^x\}_{pk_T}$, $\{(c_i^{(j)})^x\}_{pk_D}$ together with the voter's index $\{\{id_i\}\}_{pk_T}$ are being forwarded to D_k.
3. The proxy D_k verifies the proof and sends the signed value of $\{(c_i^{(j)})^x\}_{pk_D}$ back to the voter as confirmation.[4]
She further computes $\{c_i^{(j)}\}_{pk_T}$ as $\{(c_i^{(j)})^x\}_{pk_T}^{1/x}$.

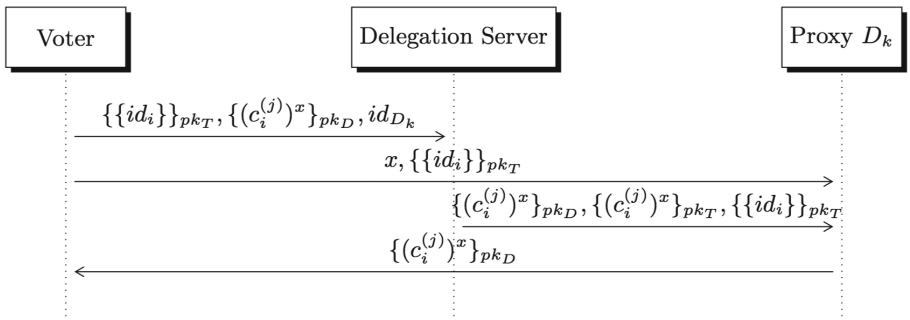


Fig. 1. Delegation of the voter id_i to the proxy D_k, with zero-knowledge proofs omitted.

Casting a Delegated Vote. To cast a delegated vote for a (unknown) voter X with (unknown) priority Y the proxy first calculates $\{\{c_X^{(Y)}\}\}_{pk_T}$. For this - given an encryption $\{c_X^{(Y)}\}_{pk_T} = (c_1, c_2)$ - she chooses a random value s and computes $\{\{c_X^{(Y)}\}\}_{pk_T} = (g_1^s, c_1, c_2 h^s)$. Then She encrypts her voting option o and prepares her ballot according to the voting process outlined above.

Cancelling a Delegation. If the voter intends to withdraw one or several vote delegations (but not excluding delegation in general), she issues the highest prioritized unused credential, overriding all previously cast votes on her behalf.

Obfuscating the Number of Cast Votes. The fake votes intended to hide the number of votes cast by a specific voter or her proxy are added accordingly to Spycher *et al.*. For each identifier id_i the tabulation tellers cast a random number of votes of the following form:

$$\langle \{\{id_i\}\}_{pk_T}, \{\{r\}\}_{pk_T}, \{\{o\}\}_{pk_T}, \sigma \rangle$$

[4] This can be done via a two-way anonymous channel or by publishing the signature on $\{(c_i^{(j)})^x\}_{pk_D}$ on the bulletin board.

Here r denotes a fake credential randomly drawn for each fake vote, a random valid voting option o, and the respective zero knowledge proofs σ. The number of fake votes for each voter is secret and is distributed as outlined in [18].

First Anonymization. After all the votes have been cast, votes with invalid proofs are removed and excluded from further tallying. Thereafter, tuples of the form $\langle \{\{id\}\}_{pk_T}, \{\{c\}\}_{pk_T}, \{\{o\}\}_{pk_T} \rangle$ are anonymized by the application of a mix net.

Ballot Clustering. After anonymizing the tuples the values of id are decrypted. For any index id_i appearing within at least one anonymized tuple the respective ordered list of credentials $(\{\{c_1^{(1)}\}\}_{pk_T}, \ldots, \{\{c_1^{(T)}\}\}_{pk_T})$ is obtained from the bulletin board. All tuples sharing the same id_i are clustered. The ordered of credentials is attached to each cluster, and the value id_i is removed from all tuples.

Second Anonymization. All lists of resulting tuples together with the respective list of attached credentials are anonymized by the application of a mix net. Note that the order of ciphertexts within the tuples is preserved so that the priority order of delegation credentials remains intact.

Extracting Votes to be Counted. In addition to weeding out the votes with invalid credentials our goal is to tally only the vote with the highest priority in case delegation took place. In order to do this for any element of the cluster - namely a list of tuples and public credentials - the list of tuples is matched on the basis of plaintext equivalence tests (PET) in decreasing order against the list of public credentials. Starting with the highest priority credential c_1 PETs are executed between all credentials of submitted tuples until a match is found. If a match is found the value $\{\{o\}\}_{pk_T}$ is extracted from the matching tuple. Once the process has been terminated for all tuples a list of encrypted voting options $(\{\{o_1\}\}_{pk_T}, \ldots, \{\{o_n\}\}_{pk_T})$ has been extracted.

Third Anonymization and Result Calculation. The list of encrypted voting options is anonymized by the application of a mix net. Eventually, the anonymized encrypted voting options are decrypted and the result is determined.

5 Security of the Proposed Scheme

In this section we consider the security evaluation of our scheme. We first summarize the security assumptions that need to be made, and then provide an informal security argument regarding the requirements given in Sect. 2.

5.1 Security Assumptions

We summarize the security assumptions specific to our scheme in the list below:

1. More than 2/3 of all tabulation tellers are trusted not to disclose private key shares to the adversary.
2. At least 1/3 of all tabulation tellers does not fail during decryption.
3. At least one of DS does not fail to forward the delegated votes to the proxy.
4. The DS does not disclose private information to the adversary.
5. The public key infrastructure for the proxies is trustworthy.
6. The private signing and designated-verifier keys of the proxy is not leaked.
7. Communication channels between the voters and the proxies are private.

5.2 Security Evaluation

We hold on to the assumptions of the original scheme which we provide in Sect. 3.2. The security properties of our scheme and the additional assumptions that are needed for them can then be evaluated as follows:

Vote Integrity. The assumptions on integrity for voters remain the same as in the original scheme.

Availability. The election results can be computed under the assumption that at least $k = \frac{1}{3}n$ tabulation tellers participate in the decryption.

Vote Secrecy for Voters. The system provides probabilistic coercion resistance, as there are two scenarios where the coercer can tell whether the voter has obeyed. The first scenario occurs if the number of fake votes for some voter equals the known minimal number. The probability of this can be controlled with the choice of an appropriate distribution function for fake votes. In the second attack, the coercer requests all the delegation credentials from the voter, and casts a vote with priority j that is unknown to the voter. In that case, unless there is a vote cast with the same priority from another voter, the coercer knows whether the credential given to her was real. The success probability of such an attack can be reduced with a smaller value of T. For example, with $T = 3$ the voter can either vote herself, delegate once or choose one back-up proxy, which we assume to be sufficient functionality for most of the elections.

Outside of these scenarios, the system provides vote secrecy, and with it coercion resistance for the voters under the same assumptions as the original scheme with $k \leq \frac{1}{3}n$. Since the delegating credentials are generated and distributed in the same way as the voting credentials the voter can cheat the coercer who acts as a proxy by providing a fake credential instead. Even if the coercer is watching the voter directly during the delegation, the voter can input a random value as her delegating credential so that the coercer cannot distinguish it from the real one.

Delegation Integrity. Casting a delegated vote without voter's permission or a delegated vote with a higher than given priority would require the knowledge of the corresponding credential $c_i^{(j)}$ for the voter i and priority j. Given that each one of those credentials is generated in the same way as the voter credentials in

the original scheme it holds that they are not leaked under the same assumptions. Another way to break the delegation integrity would be to intercept the value x used for blinding the credential forwarded to the proxy which requires control over the communication channel between voter and proxy. Furthermore, it must be assumed that the public key infrastructure for the proxies is trustworthy so that impersonation of a proxy to a voter is infeasible.

Delegation Availability. The proxy receives the credential from the DS accompanied by the designated verifier proof. In case no credential is sent the voter gets no confirmation and thus is able to repeat the delegation by choosing another DS. For this it must be assumed that the proxy's private key is not leaked and the confirmation cannot be sent by someone else. However, the DS is capable of submitting an invalid credential if it can fake the designated verifier proof. Therefore, it must be assumed that the private designated-verifier key of the proxy is not being leaked.

Vote Secrecy for Proxies. Given an anonymous channel between the proxies and the bulletin board the secrecy of votes cast by proxies corresponds to the vote secrecy for the voters. An additional assumption is required that the DS is not collaborating with the adversary. In this case using a designated-verifier proof the proxy can fake the credentials resulting from the delegation process and present them to the coercer if forced to do so.

Delegation Privacy. The proxy could be capable of learning the identity of the delegating voter in following ways: (1) by identifying the message's origin via network, (2) by learning the voter's ID, (3) by being able to assign the given delegating credential to the voter's identity. The communication channels between voters and proxies are anonymous, the voter ID is sent encrypted and only decrypted after anonymization. The delegating credentials are not otherwise leaked which is similar to the voter's credentials in the original scheme. This implies that the proxy is incapable of determining the identity of voters delegating to her. In case of a coerced proxy the coercer would not have a possibility of knowing whether the credential passed to them is valid and whether the voter has cast another vote herself. This corresponds with the coercion-resistance properties of the original scheme.

Delegation Unprovability. Given the anonymous communication channels between the DS and the voters a proxy cannot prove for given credentials that they come from actual voters and are not simulated by the proxy herself. Moreover, due to the coercion resistance properties of the original scheme and its credential generation process the proxy herself cannot verify whether the credential she received is valid. This implies that the proxy is incapable of constructing a convincing proof of possessing any amount of delegated votes.

6 Conclusion

Proxy voting constitutes a new voting mode in which voters are able to delegate their right to vote on issues beyond their expertise. At the same time, it also opens new attack vectors as there is a legitimate possibility to transfer a vote to another person who could be a coercer. To address these issues we created an Internet proxy voting scheme which focuses on coercion resistance and is based on the well-known JCJ/Civitas scheme. It provides functionality that enables vote delegation while at the same time ensuring the security of the delegation.

As our extension introduces additional credentials for delegation, which might overwhelm voters, an important part of future work would be to improve usability of the scheme. In the future, we will consider the proposal by Neumann and Volkamer [15] to address the credential handling in coercion-resistant proxy voting.

Acknowledgements. This project (HA project no. 435/14-25) is funded in the framework of Hessen ModellProjekte, financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

This paper has been developed within the project 'VALID' - Verifiable Liquid Democracy - which is funded by the Polyas GmbH.

References

1. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012)
2. Bernhard, D., Neumann, S., Volkamer, M.: Towards a practical cryptographic voting scheme based on malleable proofs. In: Heather, J., Schneider, S., Teague, V. (eds.) Vote-ID 2013. LNCS, vol. 7985, pp. 176–192. Springer, Heidelberg (2013)
3. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: pitfalls of the Fiat-Shamir heuristic and applications to helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012)
4. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
5. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
6. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. Technical report (2008)
7. Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols: a taster. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections. LNCS, vol. 6000, pp. 289–309. Springer, Heidelberg (2010)
8. Grewal, G.S., Ryan, M.D., Bursuc, S., Ryan, P.Y.: Caveat coercitor: coercion-evidence in electronic voting. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 367–381. IEEE (2013)

9. Jakobsson, M.: On quorum controlled asymmetric proxy re-encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 112–121. Springer, Heidelberg (1999)
10. Jakobsson, M., Juels, A.: Mix and match: secure function evaluation via ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
11. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
12. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70. ACM (2005)
13. Langer, L., Schmidt, A., Buchmann, J., Volkamer, M.: A taxonomy refining the security requirements for electronic voting: analyzing helios as a proof of concept. In: ARES 2010 International Conference on Availability, Reliability, and Security, pp. 475–480. IEEE (2010)
14. Neumann, S., Kahlert, A., Henning, M., Richter, P., Jonker, H., Volkamer, M.: Modeling the German legal latitude principles. In: Wimmer, M.A., Tambouris, E., Macintosh, A. (eds.) ePart 2013. LNCS, vol. 8075, pp. 49–56. Springer, Heidelberg (2013)
15. Neumann, S., Volkamer, M.: Civitas and the real world: problems and solutions from a practical point of view. In: 2012 Seventh International Conference on Availability, Reliability and Security (ARES), pp. 180–185. IEEE (2012)
16. Neumann, S., Volkamer, M.: A holistic framework for the evaluation of internet voting systems. In: Design, Development, and Use of Secure Electronic Voting Systems, pp. 76–91 (2014)
17. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
18. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 182–189. Springer, Heidelberg (2012)
19. Tchorbadjiiski, A.: Liquid democracy diploma thesis. RWTH AACHEN University, Germany (2012)
20. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 100–113. Springer, Heidelberg (2010)
21. Zwattendorfer, B., Hillebold, C., Teufl, P.: Secure and privacy-preserving proxy voting system. In: 2013 IEEE 10th International Conference on e-Business Engineering (ICEBE), pp. 472–477. IEEE (2013)

A Posteriori Openable Public Key Encryption

Xavier Bultel[1,2(✉)] and Pascal Lafourcade[1,2]

[1] CNRS, UMR 6158, LIMOS, 63173 Aubière, France
[2] Université Clermont Auvergne, LIMOS, BP 10448, 63000 Clermont-Ferrand, France
Xavier.Bultel@UDAMAIL.FR

Abstract. We present a public key encryption primitive called *A Posteriori Openable Public Key Encryption* (APO-PKE). In addition to conventional properties of public key cryptosystems, our primitive allows each user, who has encrypted messages using different public keys, to create a special decryption key. A user can give this key to a judge to open all messages that have been encrypted in a chosen time interval with the public keys of the receivers. We provide a generic efficient construction, in the sense that the complexity of the special key generation algorithm and this key size are independent of the number of ciphertexts. We give security models for our primitive against chosen plaintext attack and analyze its security in the random oracle model.

Keywords: Public-key encryption · Openable encryption · ROM · CPA

1 Introduction

Since the emergence of the Internet, email communication is accessible to anyone. Email privacy is an important computer security topic. Without public key encryption schemes, plaintext messages are sent and stored by the mail server without any protection. Fortunately, there exist many straightforward to use softwares that allow everyone to encrypt and sign emails using public key cryptography, such as the well known GnuPG[1] tool. Unfortunately, these softwares are rarely used [27], consequently encrypted emails may be considered as a suspect behavior. Hence as P. Zimmermann, the designer of PGP, said:"*If privacy is outlawed, only outlaws will have privacy*". We hope that in a near future everybody can privately exchange emails. Then our motivation is based on the following scenario, where Alice is implied in a court case. To find some clues, the judge needs to read emails that Alice has sent during a specified time period. The judge uses his power to obtain from Alice's email server all emails sent by Alice (including dates of dispatch and receiver identities). If the messages are not encrypted then the judge can read emails without relation to the investigation,

This research was conducted with the support of the "Digital Trust" Chair from the University of Auvergne Foundation.

[1] https://www.gnupg.org.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 17–31, 2016.
DOI: 10.1007/978-3-319-33630-5_2

which is a privacy violation. On the other hand, if messages are encrypted with the receiver public key then the judge can suspect Alice to hide crucial information for the investigation. Moreover, without the receivers' private keys, Alice has no solution to prove her innocence and cannot reveal his correspondence to the judge.

To solve this problem, Alice needs a mechanism to give to the judge a possibility to open all messages sent during a specified time period. Using our solution Alice can construct such a special key called an *interval-key*. With this key, the judge can only read the encrypted messages sent during this specific interval of time, because this key does not allow him to open other encrypted messages stored on the email server. Nowadays, to the best of our knowledge, there is no efficient cryptographic solution that offers such functionality to the users. The goal of this paper is to propose a practical and efficient solution to this problem.

In many public key cryptosystems, when a ciphertext is generated, it is possible to create a special key that allows a person to decrypt it, without knowing the corresponding secret key. For example, in ElGamal [13], $C = (C_1, C_2) = (g^r, g^{x \cdot r} \cdot m)$ is the ciphertext of the message m with the public key g^x and a random element r (for g a generator of G a group of prime order). Knowing the random element r, the public key of Bob g^x and the ciphertext C a third party can compute $C_2/(g^x)^r = m$ to recover the plaintext. Using this property it is possible to construct a naïve solution by giving n random elements to a third party to decrypt n ciphertexts. However, this method presents an inherent limitation when the number n is large and the user has to store all the random elements used to encrypt all the messages during an interval of time. The aim of this paper is to allow a user to construct an interval-key to decrypt several consecutive messages in a time interval where the size of the key, the stored information and the key generation complexity are constant and do not increase with the number of ciphertexts.

Contributions: We first present the notion of *Random Coin Decryptable Public Key Encryption* (RCD-PKE). The idea of RCD-PKE is that one can open a ciphertext with the secret key and also use the random coin used during the encryption to open a cipher. We show that several existing schemes in the literature satisfy this notion, *e.g.* [1,10,14]. We use the RCD-PKE property to construct a scheme that allows a user to generate an interval-key for a judge to open all the messages he sent during a period of time. This scheme, called *A Posteriori Openable Public Key Encryption* (APO-PKE), allows the judge to open all messages sent between two given dates. The number of ciphertexts is potentially infinite but the judge decryption capability is limited to the a posteriori chosen interval. It contains, like a standard public key encryption, a key generation function, an encryption function and a decryption function. It also has an extraction function that, given two ciphertexts and a secret value, generates an interval-key for the judge. Using this interval-key he can then open all messages encrypted by different public keys between the two ciphertexts for which the key has been created.

Our scheme is generic since it only relies on any IND-CPA secure RCD-PKE and hash functions.

Performances: Our scheme has reasonable encryption and decryption execution time overhead comparing to the PKE we use, because the size of ciphertexts generated by our scheme is approximately the double of the size of the PKE encryption. Moreover the generation of the interval-key, its size and the stored information are also independent of the number of messages contained in the interval of time. Finally, there is no restriction neither about the total number of generated ciphertexts nor about the number of ciphertexts in a time interval.

Security: We provide the security models to prove the security of our schemes in the Random Oracle Model (ROM). We prove that the judge colluding with some users cannot learn more than the messages for which he received the interval-key. We also show that several users cannot collude in order to learn information about plaintexts contained in an interval of ciphertexts with the judge interval-key. We also demonstrate that the judge gets the same plaintext as the one received by the owners of the secret keys. This means that it is not possible to forge fake messages that the judge can open and not the owners of the secret keys, and *vice-versa*.

Our construction allows us to use the extraction algorithm only once per judge (or per set of encrypted mails). Our security model captures this situation. It is not going against our motivation as long as we consider that two judges having an interval key in two different court cases (for the same set of mails) do not collude. To avoid this drawback, we need to reinitialize the secret values stored by a user after the generation of an interval-key, in order to be able to produce new interval-key on the next encrypted data. We leave the construction of an APO-PKE with constant interval key generation complexity and constant interval key size allowing several interval key generations for the same judge and the same set of encrypted mails as an open problem.

Related Work: Functional encryption [26] is a public-key encryption primitive that allows a user to evaluate a function on the plaintext message using a key and a ciphertext. This cryptographic primitive was formalized in [5]. It generalizes many well know cryptographic primitives such identity based encryption [4] or attribute based encryption [26]. Moreover, some schemes that evaluate an arbitrary function have been proposed in [17,18]. *A posteriori* openable encryption can be seen as a functional encryption, where all ciphertexts (resp. plaintexts) that are encrypted by one user correspond to a unique large ciphertext (resp. plaintext). Then the interval-keys allow a user to find only some parts of the corresponding plaintext. Our proposal scheme is an efficient solution for this kind of functional encryption.

Deniable encryption [7,22] is an encryption system that allows to encrypt two messages (original and hidden messages) in the same ciphertext. Using his secret key, the receiver can retrieve the original message. Using another shared secret

key, the receiver can also decrypt the hidden message. It is not possible for the sender to prove that his encryption does not contain an hidden encrypted message. In our *a posteriori* openable encryption, the judge is only convinced that the plaintext that he decrypts is the same message that the plaintext decrypted by the secret key of the receiver. This notion differs from undeniability since the judge is convinced that a message he decrypts using interval key has actually been sent and received, but does not deal with message from another channel that the given encryption system (including different way to encrypt or decrypt a message in the same ciphertext).

Some cryptographic primitives deal with time in decryption mechanism or rights delegation. *Timed-Release Encryption* (TRE), first proposed in [24], is a public key encryption where encrypted messages cannot be opened before a *release-time* chosen by the person who encrypted the messages. In this primitive, it is generally a time server that allows the receiver to decrypt the message *in the future* at a given date. Several TRE with diverse security properties have been proposed [3,8,9]. More recently, an extension of TRE, called *Time-Specific Encryption* (TSE), has been proposed in [25] and deals with time intervals. Somehow these primitive are close to our because APO-PKE allows somebody to give decryption capabilities *in the future*, after that encrypted messages has been sent. However, TRE and TSE cannot be used to achieve APO-PKE, because TRE ciphertext are intended to only one user and decryption capabilities cannot be delegated to another party. Moreover, in TRE, time of decryption capability must be chosen during the encryption phase, while in our primitive it can be chosen at any time (*a posteriori*).

It is interesting to note that some TRE possess a pre-open mechanism [21] that allows the sender to give decryption capabilities before the pre-specified release-time. In this case, a security requirement (called *binding* property) ensures that the decrypted message from the pre-open mechanism is the message decrypted by the receiver after the release-time [11]. For our primitive, we define a similar property, called *integrity*, since we require that decrypted messages using an interval key must be equal to the messages decrypted by the legitimate receivers.

Finally, *Key-Insulated Encryption* (KIE) [12,20,23] is a public key encryption primitive where messages are encrypted from a tag corresponding to a time period and a public key. At each time period corresponds a partial secret key computed from a master key and the previous partial secret key. Moreover, the public key is never changed. The motivation of this primitive is to provide secret keys that can be stored in an untrusted device without compromising the master key. Indeed, the leakage of a secret key compromises only messages received in a specified time interval, and future encryptions remain secure. In the motivation of [12], the authors give another interesting use of this primitive based on [16]. They provide a secure delegation of decryption rights in a time period. However, this type of delegation allows them to delegate decryption rights only on predefined time period. For example, if the time period corresponds to one month then right delegation cannot be restricted to the last week of a month and the

first week of the following month without revealing all messages of these two months. Moreover, delegator must give a different secret key to each time period, so the decryption keys are proportional to the number of time periods contained in the interval. Our goal is to propose decryption delegation capabilities to the sender, while KIE only focuses on receiver decryption right delegation. Thus this primitive cannot solve our problem.

Outline: In the next section, we introduce some cryptographic tools and define the notion of RCD-PKE. In Sect. 3, we present a generic *A Posteriori Openable Public Key Encryption*. Then in Sect. 4, we provide security models and analyze the security of our scheme before concluding in the last section. All the proofs of our security results are given in the full version of this paper [6].

2 Random Coin Decryptable Public Key Encryption

We first recall the definition of probabilistic public key encryption.

Definition 1 (Probabilistic Public Key Encryption (PKE)). *A* probabilistic PKE *is a triplet of polynomial time algorithms* $(\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ *such that* $\mathsf{Gen}(1^k)$ *returns a public/private key pair* $(\mathit{pk}, \mathit{sk})$, $\mathsf{Enc}_{\mathsf{pk}}(m;\sigma)$ *returns a ciphertext* c *from the public key* pk*, the message* m *and the random coin* σ*, and* $\mathsf{Dec}_{\mathsf{sk}}(c)$ *returns a plaintext* m *or a bottom symbol* $\perp$ *from a secret key* sk *and a ciphertext* c*. Moreover the following equation holds:* $\mathit{Dec}_{\mathit{sk}}(\mathit{Enc}_{\mathit{pk}}(m;\sigma)) = m$.

A PKE scheme Π is said indistinguishable under chosen-plaintext attack (IND-CPA) [19] if for any polynomial time adversary $\mathcal{A}$, the difference between $\frac{1}{2}$ and the probability that $\mathcal{A}$ wins the IND-CPA experiment described in Fig. 1 is negligible.

$\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(k)$:
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$
$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_0(1^k, \mathsf{pk})$
$c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b;\sigma)$
$b' \leftarrow \mathcal{A}_1(\mathsf{st}, \mathsf{pk}, c)$
return $(b = b')$

Fig. 1. IND-CPA experiment.

We introduce the notion of *Random Coin Decryptable* PKE (RCD-PKE). A public key encryption scheme is said RCD-PKE, if there exists a second way to decrypt the ciphertext with the random coin used to construct the ciphertext. This primitive is a kind of PKE with *double decryption* mechanism (DD-PKE) which is defined in [15]. Actually RCD-PKE is a DD-PKE where the second secret key is the random coin and is used once.

Definition 2 (Random Coin Decryptable PKE (RCD-PKE)). *A probabilistic PKE is* Random Coin Decryptable *if there exists a polynomial time algorithm* CDec *such that for any public key* pk*, any message* m*, and any coin* σ*, the following equation holds:* $\mathit{CDec}_{\sigma}(\mathit{Enc}_{\mathit{pk}}(m;\sigma), \mathit{pk}) = m$.

For instance, ElGamal encryption scheme is RCD-PKE. It is possible, from a ciphertext $c = \mathsf{Enc}_{\mathsf{pk}}(m;\sigma) = (c_0, c_1) = (g^\sigma, \mathsf{pk}^\sigma \cdot m)$ to use the algorithm $\mathsf{CDec}_\sigma(c, \mathsf{pk})$ that computes c_1/pk^σ to retrieve the plaintext message m. Many probabilistic encryption schemes in the literature are RCD-PKE, *e.g.* [1,10,14]. Algorithms CDec of these two cryptosystems PKE are given in the full version of this paper [6]. We also introduce the concepts of *valid key pair* and of *verifiable key PKE.*

Definition 3 (Verifiable Key PKE (VK-PKE)). *We say that a key pair* (pk, sk) *is* valid *for* $PKE = (Gen, Enc, Dec)$ *when for any message* m *and any random coin* σ *the equation* $Dec_{sk}(Enc_{pk}(m;\sigma)) = m$ *holds. We say that a probabilistic PKE is* verifiable-key *(VK) when there exists an algorithm Ver such that* $Ver(pk, sk) = 1$ *if and only if* (pk, sk) *is valid for PKE.*

In many probabilistic public key cryptosystems, the public key is generated from the secret key by a deterministic algorithm. For example, the ElGamal public key is the value g^x computed from the secret key x. In this case, it suffices to check that $g^{\mathsf{sk}} = \mathsf{pk}$ in order to be convinced that a key pair $(\mathsf{pk}, \mathsf{sk})$ is valid. It is easy to see that [1,10] are also VK-PKE.

3 A Posteriori Openable Public Key Encryption

An APO-PKE is a public key encryption scheme, where Alice can use receiver public keys to send them encrypted messages that can be opened thanks to the corresponding secret keys. The goal of an APO-PKE is to allow Alice to keep enough information to be able to construct a key to *a posteriori* open a sequence of messages that she had encrypted during an interval of time. We do not consider real time but a sequence of n successive ciphertexts $\{C_x\}_{1\leq x\leq n}$ that have been encrypted by Alice with possibly different public keys. Then with an APO-PKE, it is possible for Alice to *extract* a key for a judge that opens all ciphertexts between the message C_i and the message C_j where $1 \leq i < j \leq n$. We call this key an *interval-key* denoted by $K_{i\to j}^{\mathsf{pko}}$ where pko is the public key of the opener (here the judge). Moreover before encrypting her first message with a public key, Alice needs to *initialize* a *secret global state* denoted st. The goal of st is to keep all required information to generate an interval-key and to encrypt a new message. Naturally each time Alice encrypts a message with a public key, st is updated (but has a constant size). Finally an APO-PKE, formally described in Definition 4, contains an algorithm that *opens* all ciphertexts in a given interval of time thanks to the interval-key forged by Alice.

Note that all key pairs come from the same algorithm APOgen. However, for the sake of clarity, we denote by pko and sko (for *opener public key* and *opener secret key*) the keys of an interval-key recipient, *e.g.* a judge that can open some messages, denoted by O (for opener) in the rest of the paper.

Definition 4 (A Posteriori Openable Public Key Encryption (APO-PKE)). *An APO-PKE is defined by:*

$\mathsf{APOgen}(1^k)$*: This algorithm generates a key pair for a user. It returns a public/private key pair* $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{APOini}(1^k)$*: This algorithm initializes a global state* st *and returns it.*

$\mathsf{APOenc}^{\mathsf{st}}_{\mathsf{pk}}(m)$*: This algorithm encrypts a plain-text* m *using a public key* pk *and a global state* st*. It returns a ciphertext* C *and* st *updated.*

$\mathsf{APOdec}_{\mathsf{sk}}(C)$*: This algorithm decrypts a ciphertext* C *using the secret key* sk*. It returns a plaintext* m *or* $\perp$ *in case of error.*

$\mathsf{APOext}^{\mathsf{st}}_{\mathsf{pko}}(C_i, C_j)$*: This algorithm generates an interval-key* $K^{\mathsf{pko}}_{i\to j}$ *that allows the owner* O *of the public key* pko *to decrypt all messages* $\{C_x\}_{i\leq x\leq j}$ *using algorithm* APOpen.

$\mathsf{APOpen}_{\mathsf{sko}}(K^{\mathsf{pko}}_{i\to j}, \{C_x\}_{i\leq x\leq j}, \{\mathsf{pk}_x\}_{i\leq x\leq j})$*: Inputs of this algorithm contain a ciphertext set* $\{C_x\}_{i\leq x\leq j}$ *and all the associated public keys* $\{\mathsf{pk}_x\}_{i\leq x\leq j}$*. This algorithm allows a user to decrypt all encrypted messages sent during an interval using his secret key* sk *and the corresponding interval-key* $K^{\mathsf{pko}}_{i\to j}$*. It returns a set of plaintexts* $\{m_x\}_{i\leq x\leq j}$ *or* $\perp$ *in case of error.*

In Scheme 1, we give a generic construction of APO-PKE based on an IND-CPA secure RCD-PKE and three hash functions.

Scheme 1 (Generic APO-PKE (G-APO)). *Let* k *be a security parameter,* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a RCD and VK PKE scheme,* $\mathcal{R}$ *be the set of possible random coins of* $\mathcal{E}$ *and* $\mathsf{F} : \{0,1\}^* \to \{0,1\}^k$, $\mathsf{G} : \{0,1\}^* \to \mathcal{R}$ *and* $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{2k}$ *be three universal hash functions. Our generic APO-PKE is defined by the following six algorithms where* $\oplus$ *denotes the exclusive-or,* $|x|$ *denotes the bit size of message* x *and* $y||z$ *the concatenation of* y *with* z*:*

$\mathsf{APOgen}(1^k)$*: This algorithm generates* $(\mathsf{pk}, \mathsf{sk})$ *with* Gen *and returns it.*

$\mathsf{APOini}(1^k)$*: This algorithm picks three random values* $\widehat{\sigma} \xleftarrow{\$} \{0,1\}^k$, $\widetilde{\sigma} \xleftarrow{\$} \{0,1\}^k$ *and* $K \xleftarrow{\$} \{0,1\}^k$ *of the same size, and returns the state* $\mathsf{st} = (K||\widehat{\sigma}||\widetilde{\sigma})$.

$\mathsf{APOenc}^{\mathsf{st}}_{\mathsf{pk}}(m)$*: We note that* $\mathsf{st} = (K||\widehat{\sigma}_N||\widetilde{\sigma}_N)$*. This algorithm picks a random* $\widehat{m}$ *such that* $|\widehat{m}| = |m|$ *and computes* $\widetilde{m} = \widehat{m} \oplus m$*. Let* $\widehat{\sigma} \xleftarrow{\$} \{0,1\}^k$ *and* $\widetilde{\sigma} \xleftarrow{\$} \{0,1\}^k$ *be two random values of size* $|\widehat{\sigma}_N|$*. This algorithm computes* $\widehat{C} = \mathsf{Enc}_{\mathsf{pk}}(\widehat{m}||(\widehat{\sigma} \oplus \mathsf{F}(\widehat{\sigma}_N)); \mathsf{G}(\widehat{\sigma}_N))$ *and* $\widetilde{C} = \mathsf{Enc}_{\mathsf{pk}}(\widetilde{m}||(\widetilde{\sigma}_N \oplus \mathsf{F}(\widetilde{\sigma})); \mathsf{G}(\widetilde{\sigma}))$*. It also computes* $D = (\widehat{\sigma}_N||\widetilde{\sigma}) \oplus \mathsf{H}(K||\widehat{C}||\widetilde{C})$*. Finally it updates the state* st *with* $(K||\widehat{\sigma}||\widetilde{\sigma})$ *and returns* $C = (\widehat{C}||\widetilde{C}||D)$.

$\mathsf{APOdec}_{\mathsf{sk}}(C)$**:** *The decryption algorithm computes the decryption of* $\widehat{m}||\widehat{\sigma} = \mathsf{Dec}_{\mathsf{sk}}(\widehat{C})$ *and the decryption of* $\widetilde{m}||\widetilde{\sigma} = \mathsf{Dec}_{\mathsf{sk}}(\widetilde{C})$*, where* $C = (\widehat{C}||\widetilde{C}||D)$*. It returns* $m = \widehat{m} \oplus \widetilde{m}$.

$\mathsf{APOext}^{\mathsf{st}}_{\mathsf{pko}}(C_i, C_j)$**:** *Using the state* $\mathsf{st} = (K||\widehat{\sigma}_N||\widetilde{\sigma}_N)$, $C_i = (\widehat{C}_i||\widetilde{C}_i||D_i)$ *and* $C_j = (\widehat{C}_j||\widetilde{C}_j||D_j)$*, this algorithm computes* $\widehat{\sigma}_{i-1}||\widetilde{\sigma}_i = D_i \oplus \mathsf{H}(K||\widehat{C}_i||\widetilde{C}_i)$ *and* $\widehat{\sigma}_{j-1}||\widetilde{\sigma}_j = D_j \oplus \mathsf{H}(K||\widehat{C}_j||\widetilde{C}_j)$*. It picks* $r \xleftarrow{\$} \mathcal{R}$ *and returns* $K^{\mathsf{pko}}_{i\to j} = \mathsf{Enc}_{\mathsf{pko}}((\widehat{\sigma}_{i-1}||\widetilde{\sigma}_j); r)$.

$\mathsf{APOpen}_{\mathsf{sko}}(K_{i\to j}^{\mathsf{pko}}, \{(\widehat{C}_x||\widetilde{C}_x||D_x)\}_{i\leq x\leq j}, \{\mathsf{pk}_x\}_{i\leq x\leq j})$**:** *This algorithm begins to recovering values* $\widehat{\sigma}_{i-1}||\widetilde{\sigma}_j = \mathsf{Dec}_{\mathsf{sko}}(K_{i\to j}^{\mathsf{pko}})$.

- *For all* x *in* $\{i, i+1, \ldots, j\}$*, it computes* $\widehat{R} = \mathsf{G}(\widehat{\sigma}_{x-1})$ *and opens* $\widehat{C}_x$ *as follows* $\widehat{m}_x||\widehat{\sigma}_x^* = \mathsf{CDec}_{\widehat{R}}(\widehat{C}_x, \mathsf{pk}_x)$*. It computes the next* $\widehat{\sigma}_x = \widehat{\sigma}_x^* \oplus \mathsf{F}(\widehat{\sigma}_{x-1})$*. If* $\mathsf{Enc}_{\mathsf{pk}_x}((\widehat{m}_x||\widehat{\sigma}_x^*); \mathsf{G}(\widehat{\sigma}_{x-1})) \neq \widehat{C}_x$ *then it returns* $\perp$.
- *For all* x *in* $\{j, j-1, \ldots, i\}$*, it computes* $\widetilde{R} = \mathsf{G}(\widetilde{\sigma}_x)$ *and opens* $\widetilde{C}_x$ *as follows* $\widetilde{m}_x||\widetilde{\sigma}_{x-1}^* = \mathsf{CDec}_{\widetilde{R}}(\widetilde{C}_x, \mathsf{pk}_x)$*. It computes the previous* $\widetilde{\sigma}_{x-1} = \widetilde{\sigma}_{x-1}^* \oplus \mathsf{F}(\widetilde{\sigma}_x)$*. If* $\mathsf{Enc}_{\mathsf{pk}_x}((\widetilde{m}_x||\widetilde{\sigma}_{x-1}^*); \mathsf{G}(\widetilde{\sigma}_x)) \neq \widetilde{C}_x$ *then it returns* $\perp$.

Finally, it returns $\{\widehat{m}_x \oplus \widetilde{m}_x\}_{i\leq x\leq j}$.

The encryption algorithm APOenc separates the plaintext m in two parts using xor operation such that $m = \widehat{m} \oplus \widetilde{m}$. We generate two random coins $\widehat{\sigma}$ and $\widetilde{\sigma}$. Using the two previous coins $\widehat{\sigma}_N$ and $\widetilde{\sigma}_N$ in the state st, we encrypt into two different ciphertexts $\widehat{C}$ and $\widetilde{C}$ the following two messages $\widehat{m}||(\widehat{\sigma} \oplus \mathsf{F}(\widehat{\sigma}_N))$ and $\widetilde{m}||(\widetilde{\sigma}_N \oplus \mathsf{F}(\widetilde{\sigma}))$. Finally we hide the usefull random elements with $\mathsf{H}(K||\widehat{C}||\widetilde{C})$.

Knowing the secret key it is possible to recover $\widehat{m}$ and $\widetilde{m}$ and then to obtain the plaintext m thanks to the algorithm APOdec.

An interval-key for the owner O of a public key pko is constructed using the algorithm APOext. It is simply the encryption with pko of $\widehat{\sigma}_N$ and $\widetilde{\sigma}$. At each encryption, the values $\widehat{\sigma}_{i-1}$ and $\widetilde{\sigma}_i$ are masked by a *"one time pad"* with the digest $\mathsf{H}(K||\widehat{C}_i||\widetilde{C}_i)$ in D_i. Then with the ciphertexts C_i, C_j and the secret value K we can construct an interval-key that contains these values $\widehat{\sigma}_{i-1}$ and $\widetilde{\sigma}_j$.

Using an interval-key $K_{i\to j}^{\mathsf{pko}}$ it is possible to open all ciphertexts encrypted during an interval of time with the algorithm APOpen: thanks to the RCD property, someone who knows values $\widehat{\sigma}_N$ and $\widetilde{\sigma}$ for one ciphertext can open each part $\widehat{C}$ and $\widetilde{C}$ of it in order to recover $\widehat{\sigma}$ and $\widetilde{\sigma}_N$, and $\widehat{m}$ and $\widetilde{m}$, hence m. We also notice that with $\widehat{\sigma}_i$ it is possible to decrypt all ciphertexts in $\{\widehat{C}_x\}_{(i+1)\leq x\leq N}$. In the other hand, with $\widetilde{\sigma}_j$ it is possible to decrypt all ciphertexts in $\{\widetilde{C}_x\}_{1\leq x\leq j}$. Then it is possible to recover all messages between C_i and C_j. Thus, it is possible to decrypt all messages between C_i and C_j with the knowledge of $\widehat{\sigma}_{i-1}$ and $\widetilde{\sigma}_j$.

If the interval always contains the first message, we give a more efficient algorithm. The idea is to only keep one part of the ciphertext, by consequence we do not need to split into two the message m. Hence the size of the ciphertext is smaller. Similarly if the algorithm always ends with the last encrypted message, we can also drop one half of the ciphertext and the tag value following the same idea. These simpler schemes are given in the full version of this paper [6].

4 Model and Security

We present the security properties of an $\mathsf{APO\text{-}PKE}$ scheme and we analyze the security of our $\mathsf{G\text{-}APO}$ scheme. The first security property corresponds to a chosen-plaintext attack scenario where the adversary has access to interval-keys on intervals that do not contain the challenge. We next introduce the notion of *indistinguishability under chosen sequence of plaintext attack* security

(IND-CSPA) that corresponds to a chosen-plaintext attack scenario where the challenge is an interval of ciphertexts and the corresponding interval-key generated for a given judge public key. The last property is *integrity*, and captures the integrity of messages decrypted by APOpen algorithm. All security proofs are detailed in [6].

4.1 IND-CPA security

It concerns the resistance of an APO-PKE against a collusion of adversaries that have access to interval-keys in a chosen-plaintext attack scenario. For example, if we consider a judge who receives an interval-key to open a sequence of ciphertexts and who colludes with ciphertext recipients; then it ensures that they cannot deduce any information about messages that are not in the sequence. Indeed, he cannot request an interval-key for an interval containing the challenge. We define the OT-IND-CPA security when only one interval-key can be asked during the experiment. Our scheme is proved secure in this model.

Definition 5 (OT-IND-CPA Experiment). *Let Π be an APO-PKE, let k be a security parameter, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a pair of polynomial time algorithms. We define the* one-time indistinguishability under interval opener chosen-plaintext attack *(OT-IND-CPA) experiment as follows:*

$\mathbf{Exp}^{\mathbf{OT\text{-}IND\text{-}CPA}}_{\Pi,\mathcal{A}}(k)$*:*
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}_*, \mathsf{sk}_*) \leftarrow \mathsf{APOgen}(1^k)$
$\mathsf{st}_* \leftarrow \mathsf{APOini}(1^k)$
$(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_0(1^k, \mathsf{pk}_*)$
$C_* \leftarrow \mathsf{APOenc}^{\mathsf{st}_*}_{\mathsf{pk}_*}(m_b)$
$b' \leftarrow \mathcal{A}_1(\mathsf{state}, C_*)$
If $b = b'$ *return* 1*, else* 0

The adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ have access to the following oracles:

$\mathcal{O}^{\mathbf{CPA}}_{\mathbf{enc}}$**:** *On the first call to this oracle, it initializes the following values $l = 1$ and $n = 1$. This oracle takes as input a public key pk and a message m. It returns $C_l = \mathsf{APOenc}^{\mathsf{st}_*}_{\mathsf{pk}}(m)$. It increments the counter l. Only in the first phase, it increments the value n that counts the number of calls to the encryption oracle before the generation of the challenge.*

$\mathcal{O}^{\mathbf{CPA}}_{\mathbf{ext}}$**:** *The adversary can ask this oracle only one time during the experiment. This oracle takes a public key pko and two ciphertexts C'_a and C'_b. In the second phase, if there exists $C_i = C'_a$ and $C_j = C'_b$ such that $i \leq n \leq j$ then the oracle rejects the query. Else, if $C'_a = C_n$ or $C'_b = C_n$, it rejects the query. Else it returns $\mathsf{APOext}^{\mathsf{st}_*}_{\mathsf{pko}}(C'_a, C'_b)$.*

We also define the IND-CPA experiment as the same as the OT-IND-CPA experiment except that the adversary can ask the oracle APOext several times.

Definition 6 (OT-IND-CPA Advantage). *The advantage of the adversary $\mathcal{A}$ against OT-IND-CPA is defined by:*

$$\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{OT\text{-}IND\text{-}CPA}}(k) = |Pr[Exp_{\Pi,\mathcal{A}}^{\mathsf{OT\text{-}IND\text{-}CPA}}(k) = 1] - \frac{1}{2}|$$

We define the advantage on OT-IND-CPA experiment by:

$$\mathsf{Adv}_{\Pi}^{\mathsf{OT\text{-}IND\text{-}CPA}}(k) = \max\{\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{OT\text{-}IND\text{-}CPA}}(k)\}$$

for all $\mathcal{A} \in \text{POLY}(k)$. The advantages on IND-CPA experiment are similar to those of OT-IND-CPA. We say that a APO-PKE scheme Π is OT-IND-CPA (resp. IND-CPA) secure when $\mathsf{Adv}_{\Pi}^{\mathsf{OT\text{-}IND\text{-}CPA}}(k)$ (resp. $\mathsf{Adv}_{\Pi}^{\mathsf{IND\text{-}CPA}}(k)$) is negligible.

Our construction is not IND-CPA since if a judge has two interval-keys for two different intervals of time given by the same user and computed with the same secret value then he can open all messages between the two extreme dates.

Theorem 1. *Let E be an IND-CPA secure RCD-PKE, then G-APO based on E is OT-IND-CPA secure in the random oracle model.*

Proof idea: To prove the OT-IND-CPA security, we show first that no polynomial adversary wins the experiment with non negligible probability using the oracle $\mathcal{O}_{\mathsf{ext}}^{\mathsf{CSPA}}$ in an interval of previous ciphertexts of the challenge. The interval-key allows to open the part $\widehat{C}_*$ of the challenge C_*, but since the PKE is IND-CPA then the interval-key gives no information about the part of the challenge encrypted in the part $\widetilde{C}_*$. Similarly, we then prove that no adversary can win using the oracle in an interval of next ciphertexts of the challenge. Finally, using this two results, we show that our scheme is OT-IND-CPA in any case. □

4.2 IND-CSPA security

A sequence of ciphertexts coupled with an interval-key can be seen as an unique ciphertext that encrypts a sequence of plaintexts because the open algorithm allows a judge to decrypt all the messages of the sequence with the knowledge of any secret key. Thus, we define a security model where the adversary must distinguish the sequence of plaintexts used to produce a challenge sequence of ciphertexts associated to an interval-key. The IND-CSPA security captures this security property. In this model, the adversary is a collusion of users that must distinguish the sequence of plaintexts used to produce a sequence of ciphertexts given the corresponding interval-key generated for the judge.

Definition 7 (IND-CSPA$_\phi$ Experiment). *Let Π be an APO-PKE, let k be a security parameter, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be a pair of polynomial time algorithms. We define the* indistinguishability under chosen sequence of plaintext attack *(IND-CSPA$_\phi$) experiment as follows, where n denotes the number of calls to the encryption oracle during the first phase and ϕ denotes the number of calls to the generation oracle:*

$\mathbf{Exp}^{\mathsf{IND\text{-}CSPA}_\phi}_{\Pi,\mathcal{A}}(k)$:
$b, d \xleftarrow{\$} \{0,1\}$
$(pko_*, sko_*) \leftarrow APOgen(1^k)$
$st_* \leftarrow APOini(1^k)$
$(q, \{m_x^0\}_{n<x\leq n+q}, \{m_x^1\}_{n<x\leq n+q}, \{pk_x\}_{n<x\leq n+q}, state) \leftarrow \mathcal{A}_0(1^k, pko_*)$
$\forall\ x \in \{n+1, n+2, ..., n+q\}$:
  if pk_x *comes from* $\mathcal{O}^{CSPA}_{gen}$ *then* $C^*_x = APOenc^{st_*}_{pk_x}(m^b_x)$
  else, $C^*_x = APOenc^{st_*}_{pk_x}(m^d_x)$
$K^{pko_*}_{(n+1)\to(n+q)} \leftarrow APOext^{st_*}_{pko_*}(C_{n+1}, C_{n+q})$
$b' \leftarrow \mathcal{A}_1(\mathsf{state}, \{C^*_x\}_{n<x\leq n+q}, K^{pko_*}_{(n+1)\to(n+q)})$
If $b = b'$ *return* 1, *else* 0

The adversaries $\mathcal{A}_0$ *and* $\mathcal{A}_1$ *have access to the following oracles:*

$\mathcal{O}^{\mathsf{CSPA}}_{\mathsf{gen}}$**:** *At the first call, the oracle creates a keys' list* K *that contains* (pko_*, sko_*). *At each call, it generates values* (pk, sk) *from* $APOgen(1^k)$ *and adds it to* K. *Then it returns* pk. *This oracle can be called only* ϕ *times.*

$\mathcal{O}^{\mathsf{CSPA}}_{\mathsf{enc}}$**:** *This oracle takes as inputs a public key* pk *and a message* m. *Only in the first phase, it increments the value* n *that counts the number of calls to the encryption oracle before the generation of the challenge.*
In the two phases, it returns $APOenc^{st_*}_{pk}(m)$.

$\mathcal{O}^{\mathsf{CSPA}}_{\mathsf{ext}}$**:** *This oracle takes as input two ciphertexts* C_i *and* C_j. *It returns the interval-key* $K^{pko_*}_{i\to j} = APOext^{st_*}_{pko_*}(C_i, C_j)$.

In the first phase The challenger generates $(\mathsf{pko}_*, \mathsf{sko}_*)$ from $\mathsf{APOgen}(1^k)$ and a state st_* from $\mathsf{APOini}(1^k)$. He sends the public key pko_* to the adversary. The challenger initializes a counter n that counts number of calls to the oracle $\mathcal{O}^{\mathsf{CSPA}}_{\mathsf{enc}}$ during this phase. Finally, the adversary sends to the challenger values $(q, \{m^0_x\}_{n<x\leq(n+q)}, \{m^1_x\}_{n<x\leq(n+q)}, \{\mathsf{pk}_x\}_{n<x\leq n+q}, \mathsf{state})$.

In second phase, the challenger computes a sequence of ciphertexts from the adversary's output. He encrypts messages of one of the two sequences. The sequence of produced ciphertexts forms the challenge. More formally, the challenger picks two random bits b and d. Then, $\forall\ x \in \{n+1, n+2, ..., n+q\}$, if pk_x corresponds to an honest user (*i.e.* pk_x comes from oracle $\mathcal{O}^{\mathsf{CSPA}}_{\mathsf{gen}}$) then he computes $C^*_x = \mathsf{APOenc}^{\mathsf{st}_*}_{\mathsf{pk}_x}(m^b_x)$ else if pk_x corresponds to a dishonest user (*i.e.* pk_x comes from the adversary), he computes $C^*_x = \mathsf{APOenc}^{\mathsf{st}_*}_{\mathsf{pk}_x}(m^d_x)$. Finally, he computes $K^{\mathsf{pko}_*}_{(n+1)\to(n+q)} = \mathsf{APOext}^{\mathsf{st}_*}_{\mathsf{pko}_*}(C_{n+1}, C_{n+q})$ and he sends $(\mathsf{state}, \{C^*_x\}_{n<x\leq(n+q)}, K^{\mathsf{pko}_*}_{(n+1)\to(n+q)})$ to the adversary $\mathcal{A}_1$. During the guess phase, the adversary returns the bit b'. If $b' = b$ then $\mathcal{A}$ wins.

Definition 8 (IND-CSPA Advantage). *We define the advantage of* $\mathcal{A}$ *against IND-CSPA by:*

$$\mathsf{Adv}^{IND\text{-}CSPA_\phi}_{\Pi,\mathcal{A}}(k) = |Pr[Exp^{IND\text{-}CSPA_\phi}_{\Pi,\mathcal{A}}(k) = 1] - \frac{1}{2}|$$

We define by:

$$\mathsf{Adv}^{IND\text{-}CSPA_\phi}_{\Pi}(k) = \max\{\mathsf{Adv}^{IND\text{-}CSPA_\phi}_{\Pi,\mathcal{A}}(k)\}$$

for all $\mathcal{A} \in \text{POLY}(k)$ *the advantage on IND-CSPA. We say that an APO-PKE scheme* Π *is IND-CSPA secure when the advantage* $\mathsf{Adv}_{\Pi}^{\textit{IND-CSPA}_{\phi}}(k)$ *is negligible for any polynomial* ϕ.

Theorem 2. *Let* E *be a PKE that is RCD, then G-APO using* E *is IND-CSPA secure in the random oracle model.*

Proof idea: In [2] authors prove that any IND-CPA PKE is still secure in multi-user setting, *i.e.* where the adversary can ask several challenges for several different public keys. Without interval-key oracle, the IND-CSPA security of our scheme can be reduced to the IND-CPA of the PKE in multi-user setting since the challenge corresponds to ciphertexts of several messages from several public keys. Moreover, since the interval-keys from the oracle are encrypted, then the adversary must break the IND-CPA security of PKE to use it. It is possible to prove that no adversary can efficiently break the IND-CSPA of our scheme using these two arguments. □

4.3 Integrity

The last security property for APO-PKE is the *integrity*. This property is similar to *binding* property of TRE defined in [11]. The judge must be sure that the messages he decrypts with APOpen algorithm are the sent messages.

Definition 9 (Integrity Experiment). *Let* Π *a APO-PKE, let* k *be a security parameter, and let* $\mathcal{A}$ *a polynomial time algorithm. We define the* integrity *experiment as follows:*

$\mathbf{Exp}_{\Pi,\mathcal{A}}^{\textsf{Integrity}}(k)$*:*
$(\mathit{pko}_*, \mathit{sko}_*) \leftarrow \mathit{APOgen}(1^k)$
$(N, \{C_x\}_{1\leq x\leq N}, \{\mathsf{pk}_x\}_{1\leq x\leq N}, l, \mathsf{sk}_l, i, j, K_{i\to j}^{\mathsf{pko}_*}) \leftarrow \mathcal{A}(1^k, \mathsf{pk}_*)$
if $(\mathsf{pk}_l, \mathsf{sk}_l)$ *is not a valid key pair then return* 0
$\{m_x\}_{i\leq x\leq j} \leftarrow \mathit{APOpen}_{\mathsf{sko}_*}(K_{i\to j}^{\mathsf{pko}_*}, \{C_x\}_{i\leq x\leq j}, \{\mathsf{pk}_x\}_{i\leq x\leq j})$
if $m_l \neq \mathit{APOdec}_{\mathsf{sk}_l}(C_l)$ *then return* 1*, else* 0.

The challenger generates $(\mathsf{pko}_*, \mathsf{sko}_*)$ from $\mathsf{APOgen}(1^k)$ and sends the public key pko_* to the adversary. The adversary $\mathcal{A}$ sends to the challenger an integer N, an ordered set of N ciphertexts $\{C_x\}_{1\leq x\leq N}$ and an ordered set of N public keys $\{\mathsf{pk}_x\}_{1\leq x\leq N}$. The adversary then sends two integers i and j and the corresponding interval-key $K_{i\to j}^{\mathsf{pko}_*}$. He finally sends the integer l and the secret key sk_l corresponding to pk_l. If $(\mathsf{pk}_l, \mathsf{sk}_l)$ is not a valid key pair then the challenger aborts and returns 0. The challenger then computes $\{m_x\}_{i\leq x\leq j} \leftarrow \mathsf{APOpen}_{\mathsf{sko}_*}(K_{i\to j}^{\mathsf{pko}_*}, \{C_x\}_{i\leq x\leq j}, \{\mathsf{pk}_x\}_{i\leq x\leq j})$. If $m_l \neq \mathsf{APOdec}_{\mathsf{sk}_l}(C_l)$ then the challenger returns 1, else he returns 0.

Definition 10. *The advantage of* $\mathcal{A}$ *against integrity is defined by:*

$$\mathsf{Adv}_{\Pi,\mathcal{A}}^{\textit{Integrity}}(k) = Pr[Exp_{\Pi,\mathcal{A}}^{\textit{Integrity}}(k) = 1]$$

The advantage against integrity by:

$$\mathsf{Adv}_{\Pi}^{\mathit{Integrity}}(k) = \max\{\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathit{Integrity}}(k)\}$$

for all $\mathcal{A} \in \text{POLY}(k)$. *We say that a APO-PKE scheme* Π *satisfies the integrity property* $\mathsf{Adv}_{\Pi}^{\mathit{Integrity}}(k)$ *is negligible.*

Theorem 3. *Let* E *be a RCD and VK PKE that is IND-CPA secure, then G-APO using this PKE satisfies the integrity property.*

Proof idea: Since the judge has all the random coins and all the public keys used to encrypt all the opened messages, he can use them to re-encrypt these messages. Thus, if the ciphertexts that he opens correspond to the ciphertexts that he encrypts by himself, then he can conclude that the opened messages are the same as the messages decrypted by the recipient secret keys. □

5 Conclusion

We introduce the notion of RCD-PKE. Based on this notion, we propose an *a posteriori* openable PKE (APO-PKE) scheme. Our scheme allows a user to prove his innocence by showing to a judge the content of his encrypted communication with several PKE during a period of time. Our construction preserves the privacy of the others communications, meaning that the judge cannot learn any information concerning the other encrypted messages. Moreover the receivers of the encrypted messages cannot collude in order to learn more information that is contained in the received messages. Our construction is proven secure in the Random Oracle Model and is generic because it only requires RCD-PKE and hash functions.

In the future, we aim at proving that is not possible to have a secure construction that supports several generations of interval key with constant size interval-key and stored data (state). Another future work is to design a security model for chosen-ciphertext security of APO-PKE and to provide a generic construction that achieves this higher security. Finally, it may be interesting to design such a scheme in the standard model.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHIES: an encryption scheme based on the Diffie-Hellman problem. Contributions to IEEE P1363a, September 1998
2. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
3. Blake, I.F., Chan, A.C.-F.: Scalable, server-passive, user-anonymous timed release public key encryption from bilinear pairing. In: ICDS. IEEE Computer Society Press (2005)

4. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 213. Springer, Heidelberg (2001)
5. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
6. Bultel, X., Lafourcade, P.: A posteriori openable public key encryption. Technical report, University Clermont Auvergne, LIMOS (2015). http://sancy.univ-bpclermont.fr/~lafourcade/APOPKE.pdf
7. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
8. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and non-interactive timed-release encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
9. Cheon, J.H., Hopper, N.J., Kim, Y.-D., Osipkov, I.: Timed-release and key-insulated public key encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 191–205. Springer, Heidelberg (2006)
10. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1), 167–226 (2003)
11. Dent, A.W., Tang, Q.: Revisiting the security model for timed-release encryption with pre-open capability. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 158–174. Springer, Heidelberg (2007)
12. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
13. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**, 469–472 (1985)
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptol. **26**(1), 80–101 (2013)
15. Galindoa, D., Herranz, J.: On the security of public key cryptosystems with a double decryption mechanism. Inf. Process. Lett. **108**(5), 279–283 (2008)
16. Goldreich, O., Pfitzmann, B., Rivest, R.L.: Self-delegation with controlled propagation - or - what if you lose your laptop. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 153–168. Springer, Heidelberg (1998)
17. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (2013)
18. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: 45th ACM STOC, pp. 555–564. ACM Press (2013)
19. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
20. Hanaoka, G., Weng, J.: Generic constructions of parallel key-insulated encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 36–53. Springer, Heidelberg (2010)
21. Hwang, Y.-H., Yum, D.H., Lee, P.J.: Timed-release encryption with pre-open capability and its application to certified e-mail system. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 344–358. Springer, Heidelberg (2005)

22. Klonowski, M., Kubiak, P., Kutyłowski, M.: Practical deniable encryption. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 599–609. Springer, Heidelberg (2008)
23. Libert, B., Quisquater, J.-J., Yung, M.: Parallel key-insulated public key encryption without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 298–314. Springer, Heidelberg (2007)
24. May, T.: Time-release crypto. Manuscript (1993)
25. Paterson, K.G., Quaglia, E.A.: Time-specific encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 1–16. Springer, Heidelberg (2010)
26. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
27. Whitten, A., Tygar, J.D.: Why johnny can't encrypt: A usability evaluation of PGP 5.0. In: Proceedings of the 8th Conference on USENIX Security Symposium - SSYM 1999, vol. 8, p. 14. USENIX Association, Berkeley (1999)

Multicast Delayed Authentication for Streaming Synchrophasor Data in the Smart Grid

Sérgio Câmara[1(✉)], Dhananjay Anand[2], Victoria Pillitteri[2], and Luiz Carmo[1]

[1] National Institute of Metrology, Quality and Technology,
Duque de Caxias, Rio de Janeiro 25250-020, Brazil
{smcamara,lfrust}@inmetro.gov.br
[2] National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
{dhananjay.anand,victoria.pillitteri}@nist.gov

Abstract. Multicast authentication of synchrophasor data is challenging due to the design requirements of Smart Grid monitoring systems such as low security overhead, tolerance of lossy networks, time-criticality and high data rates. In this work, we propose *inf*-TESLA, Infinite Timed Efficient Stream Loss-tolerant Authentication, a multicast delayed authentication protocol for communication links used to stream synchrophasor data for wide area control of electric power networks. Our approach is based on the authentication protocol TESLA but is augmented to accommodate high frequency transmissions of unbounded length. *inf*-TESLA protocol utilizes the Dual Offset Key Chains mechanism to reduce authentication delay and computational cost associated with key chain commitment. We provide a description of the mechanism using two different modes for disclosing keys and demonstrate its security against a man-in-the-middle attack attempt. We compare our approach against the TESLA protocol in a 2-day simulation scenario, showing a reduction of 15.82 % and 47.29 % in computational cost, sender and receiver respectively, and a cumulative reduction in the communication overhead.

Keywords: Multicast authentication · Smart grid · Synchrophasors · Wide area monitoring protection and control

1 Introduction

Smart Grids are large critical cyber-physical infrastructures and are being transformed today with the design and development of advanced real-time control applications [11]. The installation of Phasor Measurement Units (PMUs) as part of world-wide grid modernization is an example of major infrastructure investments that require secure standards and protocols for interoperability [1].

PMUs take time-synchronized measurements of critical grid condition data such as voltage, current, and frequency at specific locations that are used to

S. Câmara—This work is supported in part by grants from H2020 EU-BR SecureCloud (Grant No. 2568).

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 32–46, 2016.
DOI: 10.1007/978-3-319-33630-5_3

provide wide area visibility across the grid. The synchrophasor data aggregated from multiple PMUs are used to support real-time analysis, planning, corrective actions, and automated control for grid security and resiliency. Currently, high-speed networks of PMUs are being used for Wide Area Monitoring Protection and Control (WAMPAC) applications to provide situational awareness in the Eastern and Western Interconnection of North America, in China, Canada, Brazil and across Europe [11]. Before the installation of PMUs, the lack of wide-area visibility is one of the factors that prevented early fault identification of the 2003 Northeast America and 2003 Italy blackouts [9,21]. Malicious PMU data or deliberate attacks could result in inaccurate decisions detrimental to grid safety, reliability, and security, that said, PMUs need information authentication and integrity, while confidentiality may be considered optional.

Authentication schemes in the Smart Grid must be able to efficiently support multicast. Current standard solution, suggested by IEC 62351 [5], comprises HMAC authentication algorithm for signing the synchrophasors. However, sharing only one symmetric key across a multicast group cannot guarantee adequate security, and this approach suffers from the scalability problem. The use of asymmetric cryptography and digital signatures for multicast authentication raises concerns about the impact on cost and microprocessor performance. One-Time Signature schemes can enable multicast authentication, however they suffer from communication and storage overhead, and complicated key management [24].

Although some previous literature works assume, in general, that delayed authentication is not suitable for real-time applications [7,8], such method is still eligible for some monitoring and control applications that permit relatively larger delay margins (e.g. wide-area oscillation damping control application) [25]. For more considerations on this topic, see Sect. 2. Moreover, delayed authentication presents advantages over cited issues by supporting multicast data streaming, symmetric and lightweight cryptography, corrupt data and attack detection. Also it allows scalable solutions and key management, tolerates packet loss, and provides low communication overhead and high computational efficiency.

The primary objective of this work is to propose a multicast delayed authentication protocol called *inf*-TESLA in order to provide measurement authentication in a WAMPAC application within the Smart Grid. Also, we design the Dual Offset Key Chains mechanism which is used by our protocol to generate the authenticating keys and to provide long-term communication without the need of key resynchronization between the sender and receivers. A description of two different modes for disclosing keys and a demonstration of a man-in-the-middle attack attempt against our mechanism are also provided.

Section 2 presents an overview of the network architecture used for wide area aggregation of PMU data as well as some delay constraints and authentication infrastructure. In Sect. 3 we discuss prior work in the area of packet based authentication protocols for streaming communication, and then in Sect. 4 we present the *inf*-TESLA protocol and describe the Dual Offset Key Chains mechanism along with its security properties and conditions. In Sect. 5 we evaluate our approach against the original TESLA protocol. Finally, we summarize our results and propose future works in Sect. 6.

2 Scenario Characteristics

The network architecture considered for this work is as follows. Each communication link in the infrastructure comprises one PMU sender node S capable of multicasting packets to m receivers R_k applications, where $1 \leq k \leq m$. PMU S sends time-stamped synchrophasor data packets at a rate of 10 to 120 packets per second and that can be dropped in the way to the receivers. The network has several n intermediate nodes between S and $R_k, n > 0$, called Phasor Data Concentrators (PDCs). PDCs can chronologically sort received synchrophasors as well as aggregate, repackage and route data packets to the set of higher level PDCs (Super PDCs). When packets are missing or lost, PDCs may (with due indication) interpolate measurements in order to retain the communication link.

There are different wide-area monitoring and control applications that consume synchrophasor data and have different time delays and quality requirements. For instance, Situational Awareness Dashboard, Small-Signal Stability Monitoring, and Voltage Stability Monitoring/Assessment accept up to 500 milliseconds in communication latency, other applications such as Long-term stability control, State Estimation, and Disturbance Analysis Compliance can handle up to 1000 ms. For the entire list, see [20].

Zhu *et al.* [25] simulates the latency for monitoring applications over the Smart Grid network architecture and obtained results within a range of 150–220 ms. For centralized control applications, the latency was well below 500 ms. From the delayed authentication perspective, the minimum delay of the authentication confirmation by R_k is approximately twice the latency of the network. Still, delayed authentication protocols are able to attend the requirements for the above cited applications.

When utilizing multicast communication, IEC 61850-90-5, the standard for communication networks and systems for power utility automation, requires a Key Distribution Center (KDC), which provides the symmetric key coordination between S and R_k. We assume that each S is its own KDC, which is also endorsed by the standard. Furthermore, as our scheme demands that S prove its identity to R_k once during communication initialization, each receiver is required to validate a digital signature from S and maintaining a copy of its public key certificate. For this purpose, we assume that a Public-Key Infrastructure (PKI) is also available.

2.1 Security Considerations

We assume that attacks are accordingly aligned, via a man-in-the-middle, to either manipulate data values or masquerade as a legitimate PMU. Using the attack model from [23], the adversary is not limited by network bandwidth and has full control to drop, resend, capture and manipulate packets. Although his computational resources can be large, it is not unbounded and he cannot invert a pseudorandom function with non-negligible probability. Each receiver R_k is able to authenticate both the content and source of synchrophasor payloads after a delay of d_{NMax} using our delayed authentication scheme presented in Sect. 4.

However, if a packet fails authentication at time t, then an attack that has been active and undetected since $t - d_{NMax}$ represents the maximum threat exposure.

The security primitives used throughout this paper are as follows:

- *One-way hash function* H operates on an arbitrary length input message M, returning $h = H(M)$. H can be implemented with SHA-2 family algorithms.
- *Message Authentication Code* $MAC(K, M)$ provides a tag that can verify authenticity and integrity of message M given a shared key K. $HMAC(K, M)$ is a specific construction which includes an underlying cryptographic hash function to create the authenticating tag.
- *Hash chain* $H^n(M)$ denotes n successive applications of cryptographic hash function H to message M.

3 Related Work

Multicast authentication is an active research field in recent years and has been applied to a wide range of applications. In Smart Grids, it is being used for monitoring, protection and information dissemination [24]. In this section, we review all the TESLA-based multicast authentication schemes and other multicast authentication schemes used for electrical power systems.

To address the challenge of continuous stream authentication for multiple receivers on a lossy network, Timed Efficient Stream Loss-tolerant Authentication (TESLA) was introduced by Perrig et al. [14]. Based on the Guy Fawkes protocol [2] and requiring loose time synchronization between the senders and receivers, TESLA is a broadcast authentication protocol considering delayed disclosure of keys used for authentication of previous sent messages and packet buffering by the receiver. This protocol supports fixed/dynamic packet rate and delivers packet loss robustness and scalability. Benefits of TESLA include a low computation overhead, low per-packet communication overhead, arbitrary packet loss is tolerated, unidirectional data flow, high degree of authenticity and freshness of data. Further work proposed several modifications and improvements to TESLA, allowing receivers to authenticate packets upon arrival, improved scheme scalability, reduction in overhead, and increased robustness to denial-of-service attacks [13].

Studer et al. describe TESLA++ [19], a modified version of TESLA resilient to memory-based DoS attacks. They combine TESLA++ and ECDSA signatures to build an authentication framework for vehicular ad hoc networks.

μTESLA [17] adapts TESLA to make it practical for broadcast authentication in severely resource-constrained environments; like sensor networks. Some of these adaptations include the use of only symmetric cryptography mechanisms, less frequent disclosure of keys and restriction on the number of authenticated senders. Liu and Ning [10] reduce the overhead needed for broadcasting key chain commitments and deal with DoS attacks. Their Multilevel μTESLA protocol considers different levels of key chains to cover the entire lifespan of a sensor.

Other methods include the One-Time Signatures family which gained popularity recently and is applicable to multicast authentication and also for WAMPAC applications. The author in [12] describes a one-time signature based broadcast authentication protocol based on BiBa. BiBa uses one-way functions without trapdoors and exploits the birthday paradox to achieve security and verification efficiency. Its drawbacks include a large public key and high overhead for signature generation.

HORS [18] is described by Reyzin et al. as an OTS scheme with fast signing and signature verification using a cryptographic hash function to obtain random subsets for the signed message and for verifying it, but it still suffers from frequent public key distribution. TSV [8] multicast authentication protocol generates smaller signatures than HORS and has lower storage requirement at the cost of increased computations in signature generation and verification. TSV+ [7], a patched version of TSV, uses uniform chain traversal and supports multiple signatures within an epoch. SCU [22] is a multicast authentication scheme designed for wireless sensor networks and SCU+ [7] adapts it for power systems using uniform chain traversal as well. TV-HORS [23] uses hash chains to link multiple key pairs together to simultaneously authenticate multiple packets and improves the efficiency of OTS by signing the first l bits of the hash of the message. As a downside, TV-HORS has a large public key of up to 10 Kbytes.

4 Proposed Solution

In this section, we propose *inf*-TESLA, a TESLA based scheme. At first, we review TESLA to give some background and then present our scheme.

4.1 TESLA

Timed Efficient Stream Loss-tolerant Authentication (TESLA) [13–16] is a broadcast authentication protocol with low communication and computation overhead, tolerates packet loss and needs loose time synchronization between the sender and the receivers.

TESLA relies on the delayed disclosure of symmetric keys, therefore the receiver must buffer the received messages before being able to authenticate them. The keys are generated as an one-way chain and are used and disclosed in the reverse order of their generation. At setup time, the sender must first set n as the index of the first element K_n. For generating the key chain, the sender picks a random number for K_n and using a pseudo-random function f, he constructs the one-way function $F : F(k) = f_k(0)$. So, the sender generates recursively all the subsequent keys on the chain using $K_i = F(K_{i+1})$. By that, the last element of the chain is $K_0 = F^n(K_n)$, and all other elements could be calculated using $K_i = F^{n-i}(K_n)$.

Each K_i looks pseudo-random and an adversary is unable to invert F and compute any K_j for $j > i$. In the case of a lost packet containing K_i, a receiver

can calculate K_i given any subsequent packet containing K_j, where $j < i$, since $K_j = F^{i-j}(K_i)$. As a result, TESLA tolerates sporadic packet losses.

The stream authentication scheme of TESLA is secure as long as the security condition holds: A data packet P_i arrived *safely*, if the receiver can unambiguously decide, based on its synchronized time and maximum time discrepancy, that the sender did not yet send out the corresponding key disclosure packet P_j.

TESLA also supports both communication with fixed or dynamic packet rate. For fixed rate, the sender discloses the key K_i of the data packet P_i in a later packet P_{i+d}, where d is a delay parameter set and announced by the sender during setup phase. The sender determines the delay d according to the packet rate r, the maximum tolerable synchronization uncertainty δ_{tMax} and the maximum tolerable network delay d_{NMax}, setting $d = \lceil (\delta_{tMax} + d_{NMax})r \rceil$. In this mode, the scheme can achieve faster transfer rates. For dynamic rate, the sender pick one key per time interval T_{int}. Each key is assigned to a uniform interval of duration T_{int}, T_0, T_1 ,..., T_n, that is, key K_i will be active during the time period T_i. The sender uses the same key K_i to compute the MAC for all packets which are sent during T_i, on the other hand, all packets during T_i disclose the key $K_{i-d'}$. In this case, $d' = \lceil (\delta_{tMax} + d_{NMax})/T_{int} \rceil$. We use the designation d and d' for fixed and dynamic rates respectively.

For each new receiver that joins the communication network, the sender initially creates an authenticated synchronization packet. This packet contains parameters such as interval information, the disclosure lag and also a disclosed key value - which is a commitment to the key chain. The sender digitally signs this packet to each new receiver before starting the streaming communication.

4.2 *inf*-TESLA

inf-TESLA, short for infinite TESLA, is a multicast authentication protocol based on TESLA suitable for use in long term communication at high packet rates. As in TESLA, *inf*-TESLA relies on the strength of symmetric cryptography and hash functions and on the delayed disclosure of keys as a means to authenticate messages from the sender. Also, it requires only loose time synchronization between the sender and the receiver and can operate under both dynamic and fixed packet rates.

By using fixed packet rate mode, there is no need for setting specific time intervals for MACing and disclosing keys. Each autheticating key is used once for the actual message and disclosed d packets later. Although this operational mode can achieve maximum speed on authenticating previous packets, it has a drawback of quickly consuming the authenticating key chain, depending on the frequency of the packets.

Since we use one-way hash functions to build independent key chains, every time one of the key chains comes to an end (meaning that it was fully used in the authentication process) the sender must automatically build, store and utilize a new key chain in its place. In the original TESLA protocol, a sender would have to reassign a new synchronization packet as the current key chain

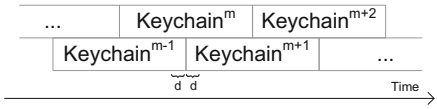


Fig. 1. An illustration of dual offset key chains as used for *inf*-TESLA.

comes to an end, inflicting non-negligible network and computational overhead by digitally signing a synchronization packet at the end of each key chain.

inf-TESLA addresses this issue by using the Dual Offset Key Chains mechanism. This mechanism uses a pair of keys for each message and guarantees continuity of the multicasting authentication process without the need for signing and sending a new synchronization packet. The mechanism creates two offset key chains so that a pair of active key chains are always available and, as the main principle, a key chain m always straddles the substitution of key chain $m-1$ with $m+1$. Figure 1 illustrates the Dual Offset Key Chains mechanism by which key chain m supports the substitution of key chain $m-1$ for key chain $m+1$ without the need for resynchronization. A detailed description of the Dual Offset Key Chains mechanism is presented on Sect. 4.2.

The overall initialization setup is similar to TESLA. Before the data streaming begins, the sender first determines some fundamental information about the network status, d_{NMax}), and time synchronization, δ_{tMax}, and builds its first two key chains. We assume that both sender and receiver are time synchronized by a reliable time protocol (e.g. PTP). After that, the sender S chooses the delay parameter d (Sect. 4.1) that will base the decision of the receiver R_k to either accept a packet from S. This condition is **Security Condition-1** for *inf*-TESLA.

For bootstrapping each new receiver, S constructs and sends the synchronization (commitment) packet to the new incomer. For a dynamic packet rate, this packet contains the following data [13]: the beginning time of a specific interval T_j along with its id I_j, the interval duration T_{int}, the key disclosure delay d', a commitment to the key chain K_i^m and key chain K_i^{m+1} ($i < j - d'$ where j is the current interval index).

For a fixed packet rate r, let j_1 and j_2 be the current key from key chains m and $m+1$ respectively. The synchronization packet contains: delay d and the commitment for the key chains $K_{i_1}^m$ and $K_{i_2}^{m+1}$ ($i_1 < j_1 - d$ and $i_2 < j_2 - d$). We will focus on fixed packet rate in this paper for the sake of brevity and convenience of notation. While a fixed packet rate is potentially more likely for the streaming applications we address, our approach is compatible with both dynamic and fixed rates.

Dual Offset Key Chains Mechanism. The Dual Offset Key Chains mechanism enables continuity in streaming authentication without the periodic resynchronization between S and $R_k \in R$ required by TESLA. Two key chains, offset in alignment, are used simultaneously by the mechanism to authenticate

messages. For every packet, there are always two active key chains and, from each chain, one non-used key available for MACing.

For constructing the two key chains, first the sender chooses n, the total number of elements on a single key chain. Let l^m be the current number of remaining elements on the key chain m. Here we assume that all created keys are deleted just after being used for authenticating messages. Let M be the maximum available memory for storing the key chains, assuming that M is big enough for storing two key chains, m and $m+1$, at any time. The value of n must be chosen accordingly to the following constraints: (i) $n \geq l^{m-1} + 2(d+1)$ and (ii) $n \leq \frac{M}{2} + d$.

The first constraint sets a minimum value for n, that is the minimum initial size of a key chain. During the initialization setup of the first receiver synchronization, we consider $l^{m-1} = 0$ for constructing the first key chain. The second constraint restricts the maximum number of elements in a key chain. If a key chain m does not meet this limit, key chain $m+1$ will not be long enough to meet the security condition for the key chain exchange procedure (see Sect. 4.2). In practice, it may not be feasible to calculate a whole key chain in the time taken to send two data packets and so S may compute and store key chain $m+1$ well before the end of key chain $m-1$.

A packet P_j sent by S is formed by the following data $P_j = \{M_j, i_1, i_2, K^m_{i_1-d}, K^{m+1}_{i_2-d}, MAC(K^m_{i_1}||K^{m+1}_{i_2}, M_j)\}$. Every packet carries the actual message M_j, the current sequence number of each key chain i_1 and i_2, the disclosed authenticating keys $K^m_{i_1-d}$ and $K^{m+1}_{i_2-d}$ (discussed later in Sect. 4.2) and the MAC of the message resultant from an operation that uses the concatenation of current keys from both key chains. In particular, at the beginning of a key chain $m+1$, the notation $K^{m+1}_{i_2-d}$ may refer to the last keys in the key chain $m-1$.

Disclosure of Keys. *inf*-TESLA has two modes of operation for disclosing keys: **2-keys** and **Alternating**. In the 2-keys mode (or standard mode, as previously described), each packet P_j discloses two authentication keys, one from each key chain, for the same message M_i, that is packet P_j has the following information, $P_j \rightarrow K^m_{i_1-d}, K^{m+1}_{i_2-d}$.

The Alternating mode discloses one key from each key chain alternatively in each data packet. Formally, two consecutive packets would have the following information about keys, $P_j \rightarrow K^m_{i_1-d}$ and $P_{j+1} \rightarrow K^{m+1}_{i_2+1-d}$, where indexes $i_1 - d$ and $i_2 - d$ correspond to the keys of both key chains to be disclosed in the same data packet in 2-keys mode of operation. Figure 2 shows the key chains in time and the two modes for disclosing keys. In Sect. 5, we present a more detailed comparison of these two modes in relation to communication overhead, computational cost and authentication delay.

The disclosure delay d for the keys is directly affected by the maximum tolerable network delay d_{NMax}, so each receiver R_k will present a different delay value. Sender S must set d as the largest expected delay in order to meet security condition-1.

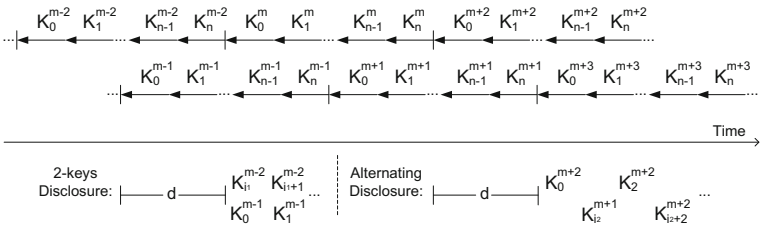


Fig. 2. Two modes for disclosing keys: 2-keys and alternating.

Dual Offset Key Chains Mechanism Security. Key chain security is based on the widely used cryptographic primitive: the one-way chain. One-way chains were first used by Lamport for one-time password [6] and has served many other applications in the literature.

The **Security Condition-2** for *inf*-TESLA concerns the key chain exchange procedure. This condition states that both key chains cannot be substituted within a time interval d/r (or within d packets). If this happens, the receiver must drop the following packets and request for resynchronization with the sender. This protocol restriction assures the authentication inviolability of *inf*-TESLA and must be observed at all times by the receiver. The receiver is solely responsible for monitoring the key chain exchange procedure and accepting, or rejecting, the new key chain.

In Fig. 3, we show an example of a man-in-the-middle attack attempt on the Dual Offset Key Chains mechanism and the importance of the security condition-2. For this example, we consider $d = 9$ as minimum number packets the sender has to wait to disclose a key, the last element $n = 50$ for all key chains, and the asterisk symbol indicates an item maliciously inserted by the attacker. The packets are presented without indices "i" for cleaner presentation.

We first illustrate how this attack can work on a single key chain mechanism without commitment packets as follows: When the attacker senses a change in the key chain by testing every disclosured key *(a)*, he inserts M_0^* as the first manipulated message and MACs it using the first element K_0^* of a forged key chain of his own. The attacker continues faking the messages and its MACs till the last authentic key used for MACing is disclosured. After that point, the attacker is able to take complete control of the communication without being detected (b). For the second part of Fig. 3, the same attack is attempted against our mechanism. Also the attacker is able to sense when a disclosured key chain comes to an end and can also substitute the messages and the MACs in the packets. However, when he tries to take complete control of the key chain by forcing the forged key K_0^{**} over the key chain $m = 2$, this indicates for the receiver a violation of the security condition-2 for the key chain exchange procedure.

Another concern is how many consecutive packets could be lost by the receiver without actually being an attack. Following the security condition for key chain substitution, there must not be two different key chain substitutions

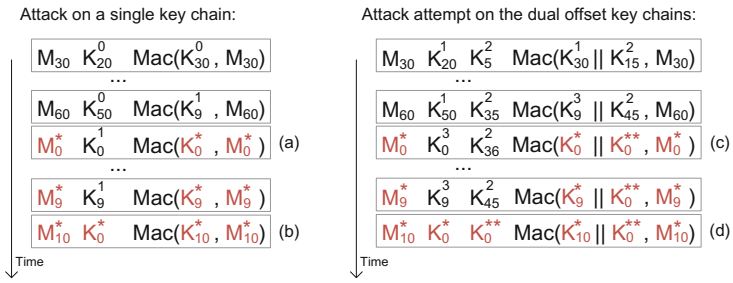


Fig. 3. Example of a man-in-the-middle attack on a single hash chain without commitment and on the Dual Offset Hash Chains mechanism.

within a period of d/r, so the receiver must be aware that the limit for consecutive packets lost is at maximum d. If, for some reason, more than d packets are lost/dropped, the receiver must assure that the following disclosed keys are authentic elements of at least one of the existing key chains, otherwise the receiver will not be able to authenticate any of the next received packets. From this point, the receiver must refuse this stream and request for a new synchronization with the sender.

Another security issue can occur when the last key K_n in the key chain's sequence is lost, that can cause a total lack of authentication of a previous packet P_n. When some K_i is lost, it can be computed from any subsequent key in the key chain through function F (Sect. 4.1), however when $i = n$ there is no subsequent key. This issue can be extended for the last d elements of the key chain, meaning that in this scenario some packets may not be authenticated and then must be dropped by the receiver. For the Alternating mode for disclosing keys, the receiver would drop $d+1$ packets in the worst case. This issue concerning the last keys of the key chain is a vulnerability of the original TESLA as well.

Elaborate attacks, like selective drop of packets, can cause even more authentication delay without being noticed. For instance, in the case of the Alternating keys disclosure mode, one attacker can induce an alternating drop of packets preventing the sender to authenticate some sequential packets. To mitigate these attacks, the receiver must set an upper limit for the maximum number of non authenticated packets to ignore before resynchronizing with the sender.

5 Evaluation Against TESLA

For the following comparison evaluation, we check for communication overhead, authentication delay and computational cost on a long term communication for each of the following schemes: original TESLA, *inf*-TESLA 2-keys (two disclosured keys per packet) and *inf*-TESLA alt (alternating key chain disclosure). Due to PMUs' operational settings, we are only considering a fixed packet rate mode. Also, we assume the following constraints for the simulation:

- Phasor data frame size of 60 bytes, according to the C37.118 standard [25], over UDP transport layer protocol.
- HMAC function and f function implementation as HMAC-SHA-256-128. Both HMAC tag size and key size of 128 bits (truncated).
- Digital signature implemented as ECDSA over GF(p) of 256 bits. Although TESLA considers RSA signatures, for comparison purposes we use ECDSA. The keys and signatures sizes are based on the NIST SP 800-131A [3] for recommendations on use of cryptographic algorithms and key lengths.
- Maximum number of keys n that can be stored at a time in the cache memory of a device is 10,000 keys.
- Sender's packet rate (frequency) of 60 packets/sec.
- Simulation time of 2 days. Past references [7] established a baseline of 1024 key chains for evaluating OTS multicast schemes. However, as *inf*-TESLA must build approximately 4 times the number of key chains as TESLA for the same number of packets, comparisons are done for fixed simulation duration rather than number of key chains. Still, for the given constraints, TESLA needs an approximate number of 1024 key chains to operate.

Table 1. Communication overhead.

	Formula
TESLA (fixed)	$C * (sKey + sSig) + P * (sKey + sMac)$
Inf-TESLA 2-keys	$2 * sKey + sSig + P * (2 * sKey + sMac)$
Inf-TESLA alt	$2 * sKey + sSig + P * (sKey + sMac)$
	2-day simulation (MBytes)
TESLA (fixed)	331,825
Inf-TESLA 2-keys	497,664
Inf-TESLA alt	331,776

Table 1 shows the formulas to calculate all security related communication overhead of each of the 3 schemes. Let C be the number of commitments (signed packets), P the total number of transmitted packets and $sKey$, $sMac$ and $sSig$ be the size of a cryptographic key, the size of the MAC tag and the size of a signature tag respectively. *inf*-TESLA 2-keys presents the higher communication overhead due to two disclosed keys per packet, while TESLA and *inf*-TESLA alt present a slightly, but negligible, difference on the overhead during two days of operation.

For calculating the computational cost overhead of each scheme, we use the formulas shown in Table 2. The processing cost in cycles per each operation of hashing, macing, signing and verifying is represented by $cHash$, $cMac$, $cSig$ and $cVer$ respectively. From the graph in Fig. 4, we can observe the higher computational cost of the sender and receiver operating TESLA over *inf*-TESLA, due to constant signing and verification operations.

Table 2. Computational cost calculation.

	Sender
TESLA (fixed)	$C * cSig + P * (cMac + cHash)$
Inf-TESLA (both)	$cSig + 2 * P * (cMac + cHash)$
	Receiver
TESLA (fixed)	$C * cVer + P * (cMac + cHash)$
Inf-TESLA (both)	$cVer + 2 * P * (cMac + cHash)$

For two days of simulation in this configuration, a sender running TESLA protocol on fixed packet rate mode has to sign up to 1036 commitment packets and spends on average 0.373117 gigacycles/hour, while running *inf*-TESLA he would spend 0.314087 gigacycles/hour of operation, which means a reduction of 15.82 % in computational cost for the sender. On the receiver side, a TESLA receiver spends in average 0.596289 gigacycles/hour, while *inf*-TESLA needs 0.314303 gigacycles/hour, meaning a reduction of 47.29 % in computational cost for the receiver. All values of cycles/operation of the security primitives are referenced from the Crypto++ Library 5.6.0 Benchmarks [4].

Although the alternating keys disclosure mode showed good results on the two previous evaluations, this mode increases the authentication delay of a packet P_i by one packet. That is because the second key needed for authenticating P_i, i.e. $K_{i_2}^{m+1}$, will only be disclosed on P_{j+1} where $j > i + d$. Also, if P_{j+1} happens to be lost, the authentication of P_i will be only achieved when receiver has the disclosed key included in P_{j+3}. On both other schemes, the authentication of a packet P_i is normally achieved after receiving P_j, $j > i + d$, and if P_j is lost, the missing keys can be recovered from the contents in P_{j+1}. Also regarding

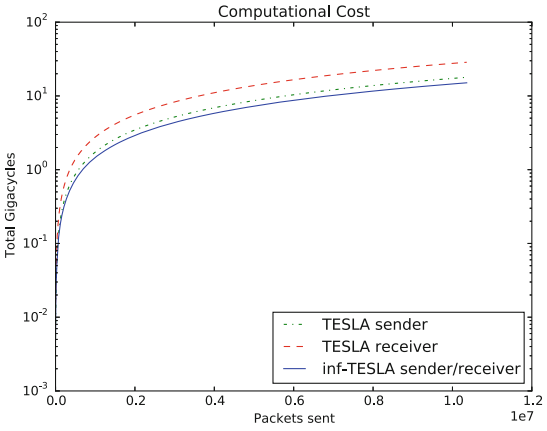


Fig. 4. Computational cost for TESLA and *inf*-TESLA over 2 days of streaming data.

authentication delay evaluation, necessary time overhead for generation and verification of digital signatures during key chains exchange may affect TESLA's continuous flow on higher frequencies of streamed data.

Although TESLA protocol is an efficient protocol and has low security overhead, it was not originally designed for long-term communication at high packet rates. We observe that *inf*-TESLA, in alternating disclosure mode, can deliver a slightly lower communication overhead and, for both modes, result in a significant reduction in computational overhead over the original protocol for the given conditions. In general, *inf*-TESLA scheme also provides great suitability for key storage and computational constrained devices, such as in Wireless Sensor Networks (WSNs).

6 Conclusion

In this work, we present *inf*-TESLA, a multicast delayed authentication protocol for streaming synchrophasor data in the Smart Grid, suitable for long-term communication and high data rates scenarios. To authenticate messages from the sender, *inf*-TESLA uses two keys to generate the MAC of the message and discloses both keys after a time frame d/r, on a fixed packet rate of operation.

We also design the Dual Offset Key Chains mechanism to produce the authenticating keys and provide a long-term communication without the need of frequently signing resynchronization packets containing commitments to the new key chains, which ensures continuity of the streaming authentication. We prove our mechanism against a man-in-the-middle attack example and describe the security conditions that must be observed at all times by the receiver. *inf*-TESLA enables two different modes for disclosing keys, 2-keys (or standard) and Alternating keys. We present a comparison between this two modes against TESLA within a WAMPAC application, and our protocol shows even more efficiency when compared to the original. Although the Alternating key disclosure mode increases the authentication delay by one packet, it provides less impact on communication overhead and a reduction of 15.82 % and 47.29 %, sender and receiver respectively, in computational cost during operational time. Generally, *inf*-TESLA shows promise and suitability for key storage and computational constrained devices.

In future work, we intend to do a further analysis on the trade-off between key storage size in devices and protocol performance, and on the possible (minimum/maximum/average) values for the authentication delay by simulating our protocol in a WAMPAC network.

References

1. Greer, C., et al.: NIST Framework and Roadmap for Smart Grid Interoperability Standards. Technical report, NIST (2014)
2. Anderson, R., Bergadano, F., Crispo, B., Lee, J.H., Manifavas, C., Needham, R.: A new family of authentication protocols. ACM SIGOPS Operat. Syst. Rev. **32**, 9–20 (1998)

3. Barker, E., Roginsky, A.: Recommendation for transitioning the use of cryptographic algorithms and key lengths. In: SP 800–131A Transitions (2011)
4. Dai, W.: Crypto++ 5.6. 0 benchmarks. Website at (2009). http://www.cryptopp.com/benchmarks.html
5. International Electrotechnical Commission: IEC TS 62351-1 Power systems management and associated information exchange - Data and communications - Part 1:Communication network and system security-Introduction to security issues (2007)
6. Lamport, L.: Password authentication with insecure communication. Commun. ACM **24**(11), 770–772 (1981)
7. Law, Y.W., Gong, Z., Luo, T., Marusic, S., Palaniswami, M.: Comparative study of multicast authentication schemes with application to wide-area measurement system. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, p. 287 (2013)
8. Li, Q., Cao, G.: Multicast authentication in the smart grid with one-time signature. IEEE Trans. Smart Grid **2**, 686–696 (2011)
9. Liscouski, B., Elliot, W.: Final report on the August 14, 2003 blackout in the United States and Canada: Causes and recommendations. A report to US Department of Energy 40(4) (2004)
10. Liu, D., Ning, P.: Multilevel μTESLA: Broadcast authentication for distributed sensor networks. ACM Trans. Embed. Comput. Syst. **3**, 800–836 (2004)
11. Patel, M., Aivaliotis, S., Ellen, E.: Real-time application of synchrophasors for improving reliability. NERC Report, October 2010
12. Perrig, A.: The BiBa one-time signature and broadcast authentication protocol. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, p. 28 (2001)
13. Perrig, A., Canetti, R., Song, D.: Efficient and secure source authentication for multicast. In: Proceedings of the Internet Society Network and Distributed System Security Symposium, pp. 35–46 (2001)
14. Perrig, A., Canetti, R., Tygar, J.D., Song, D.: Efficient authentication and signing of multicast streams over lossy channels. Proc. IEEE Symp. Secur. Priv. **28913**, 56–73 (2000)
15. Perrig, A., Canetti, R., Tygar, J., Song, D.: The TESLA broadcast authentication protocol. CryptoBytes Summer/Fall **5**, 2–13 (2002)
16. Perrig, A., Song, D., Canetti, R., Tygar, J., Briscoe, B.: Timed efficient stream loss-tolerant authentication (TESLA): Multicast source authentication transform introduction. Int. Soci. RFC **4082**, 1–22 (2005)
17. Perrig, A., Szewczyk, R., Tygar, J., Wen, V., Culler, D.E.: Spins: Security protocols for sensor networks. Wireless Netw. **8**(5), 521–534 (2002)
18. Reyzin, L., Reyzin, N.: Better than BiBa: Short one-time signatures with fast signing and verifying. Inf. Secur. Priv. **2384**, 1–47 (2002)
19. Studer, A., Bai, F., Bellur, B., Perrig, A.: Flexible, extensible, and efficient VANET authentication. J. Commun. Netw. **11**, 574–588 (2009)
20. Tuffner, F.: Phasor Measurement Unit Application Data Requirements. Technical report, Pacific Northwest National Laboratory (2014)
21. UCTE: Final Report of the Investigation Committee on the 28 September 2003Blackout in Italy. Technical Report April, Union for the Coordination of the Transmission of Electricity (2004)
22. Ugus, O., Westhoff, D., Bohli, J.M.: A rom-friendly secure code update mechanism for wsns using a stateful-verifier τ-time signature scheme. In: Proceedings of the Second ACM Conference on Wireless Network Security, pp. 29–40. ACM (2009)

23. Wang, Q., Khurana, H., Huang, Y., Nahrstedt, K.: Time valid one-time signature for time-critical multicast data authentication. In: Proceedings - IEEE INFOCOM, pp. 1233–1241 (2009)
24. Wang, W., Lu, Z.: Cyber security in the Smart Grid: Survey and challenges. Comput. Netw. **57**(5), 1344–1371 (2013)
25. Zhu, K., Nordstrom, L., Al-Hammouri, A.: Examination of data delay and packet loss for wide-area monitoring and control systems. In: 2012 IEEE International Energy Conference and Exhibition (ENERGYCON), pp. 927–934, Sept 2012

Human Aspects of Security

Developing a Human Activity Model for Insider IS Security Breaches Using Action Design Research

Gurpreet Dhillon[1], Spyridon Samonas[2(✉)], and Ugo Etudo[1]

[1] Virginia Commonwealth University, Richmond, VA, USA
{gdhillon, etudouo}@vcu.edu
[2] California State University Long Beach, Long Beach, CA, USA
ssamonas@gmail.com

Abstract. Insider security breaches in organizations have been identified as a pressing problem for academics and practitioners. The literature generally addresses this problem by focusing on the compliance of human behavior to stated policy or the conformance with organizational culture. The cultural stance and resultant activities of organizational insiders are key determinants of information security. However, whilst compliance with security policies and regulations is of great importance, the very structure of human activities that facilitates or hinders such compliance have seldom appeared in the literature. In this paper we present a human activity model that captures different aspects of a security culture. The model elucidates the patterns of behavior in organizations. Applying the model before and after an insider security breach allows us to make salient, critical areas that need attention.

Keywords: Human activity model · Insider threats · Security culture · Action design research

1 Introduction

Cyber-security threats are not always embedded in the technical infrastructure and operations of the enterprise. Vulnerabilities often arise in the socio-technical space of norms, beliefs and shared models of reality that embody an organization's culture. Developing analytics that incorporate the socio-technical environment of the organization is technically and conceptually complex.

The problem of increased security breaches that are instigated by insiders was first recognized by Denning [1]. In later years, issues related with insider threats were further vocalized through a series of reports by the now defunct Audit Commission in the UK [2–4] and the Office of Technology Assessment in the US [5]. These studies highlighted the importance and the role played by insiders in perpetuating computer related crimes. Subsequently, several case studies on Barings Bank, Kidder Peabody and Societe Generale provided evidence that, indeed, the majority of computer related threats reside within organizations (see [6]). Similar evidence has been provided by other research organizations, such as the Computer Security Institute and the CERT division of the Software Engineering Institute at Carnegie Mellon University.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 49–61, 2016.
DOI: 10.1007/978-3-319-33630-5_4

Insider threats mainly refer to the intent of dishonest employees to commit some form of cyber-crime [10, 11]. Insiders are capable of disrupting operations, corrupting data, infiltrating sensitive information, or generally compromising an IT system, thereby causing loss or damage [12, 13]. As the discussion on the insider threat has evolved considerably over the past decade, so has the very definition of the term 'insider'. Nowadays, it is not only employees who have privileged access to the assets of an organization, but also volunteers, consultants and contractors [14, 15]. Access is also given to business partners or fellow members of a strategic alliance, whereas contractors now include employees of a cloud service provider [16]. Hence, a more appropriate alternative to the term 'insider' would be a '(person with) specialized access' [16].

In this paper, we build on the Action Design Research (ADR) concept as proposed by Sein et al. [7] to develop a method for modeling insider security threats. This is undertaken in the context of an online business – Crown Confections. Our model is based on an in-depth understanding of human activities at the workplace and aims to capture both malicious and non-malicious insider threats. We draw on the organizational studies literature to argue that human activities and the *silent messages* that are emanated by individuals can be used as key indicators of insider cyber-security breaches. The application of E. T. Hall's theory of *silent messages* allows us to identify any possible trouble areas with respect to insiders and prioritize cyber-security interventions that could take the form of strategy, training or policy initiatives.

The paper adopts the publication schema for a design science research study as suggested by Gregor and Hevner [41] and is organized as follows. The following section presents a review of the literature on insider threats. The third section examines relevant literature on ADR and outlines the different stages of the method. In the fourth section, we sketch our conceptual basis for developing a human activity model of insider threats at Crown Confections. The fifth section discusses the evaluation of the model. The sixth section paper presents an analysis of the key findings, and the final section includes a summary of our research and its limitations, as well as directions for future research.

2 Literature Review

In the insider threat literature, three main informing bodies can be identified. First, studies that present ever so sophisticated technical approaches to manage insider threats. Second, studies that examine the socio-organizational aspects of insider threats. Third, studies that address different response mechanisms following an insider breach.

In the realm of technical approaches to insider threats, Denning's [1] intrusion-detection model was one of the earlier works to identify attempted system break-ins by outsiders, as well as abuse by insiders who misuse their privileges. In subsequent years several other similar models have been proposed. For instance, Bowen [6] define a threat model designed to deceive, confuse and confound attackers so that they ultimately abandon the impregnation efforts. This is undertaken by introducing host and network sensors in the architecture. Yet in other cases, honeypot technologies have been proposed as a means to catch insider threats [17]. Insider

threats within database systems have also been well researched, where user tasks are mapped to the transactions (see [18]). Generally speaking, the wealth of research in technical approaches to insider threats has focused on restricting access, technical monitoring or, at most, predictive modeling. While important, an exclusive focus on these approaches falls short of providing a holistic perspective in modeling and managing insider threats.

Socio-organizational aspects of insider threats have also been well researched (see [11] and [16]). Although the problem of the insider threat is defined and framed in different ways, and this has resulted in an abundance of approaches, the majority of studies share one thing in common: they treat all insiders as potentially malicious. In this strand of literature, insider threats emanate from available opportunities, work situations and personal factors [10]. Based on monitoring and profiling, this literature primarily addresses issues of policy compliance, sanctions, deterrence and neutralization techniques [11, 19–22]. Non-malicious violations are also examined in the insider threat literature [23]. These are typically associated with lapses, errors and mistakes, which are unintentional [24, 25]. However, more recent studies transcend the distinction between malicious and non-malicious insiders, and consider the insider threat as the departure of human behavior from compliance with security policies, irrespective of whether this is the result of malice or disregard for such policies [26, 27].

Research that examines the response to insider threats has largely been related to aspects of compliance with stated policy. As a result, the focus has been more on deterrence and the disciplinary actions that might influence human behavior. Several studies have focused on identifying antecedents to cyber-security compliance. For instance, Vance [28] found that routine activities of employees along with past behavior have a significant effect on compliance behavior. Herath [29] also found social influence to have a strong impact on compliance intention. In a related piece of research, Herath [22] investigated the role of penalties in encouraging positive information security behavior. Their findings indicate that certainty of detection has a positive impact on compliant behavior, whereas severity of punishment has a negative impact on intention to comply.

The lack of consensus in research regarding compliance antecedents has led to an exploration of other factors that might help achieve security compliance as well as ensure an adequate response to insider threats. Prominent amongst these strands of research is the work of Ramachandran [30] on security culture, Hedström [31] on value conflicts, Talib [32] on employee emancipation. Along similar lines, Dhillon et al. [33] suggest the modeling of silent messages as a way of monitoring insider threats. In this paper we heed to these assertions and provide a human activity model for insider threats.

3 Method

ADR helps in developing a prescriptive design artifact in an organizational setting. Typically, ADR is used to address a particular organizational problem through intervention and evaluation. This is followed by the construction and evaluation of the artifact, which would allow for the problem to be solved. Over the years, ADR has

been extended to areas such as business intelligence and security breach detection. However, despite its significant contributions to knowledge creation in IS, both in the form of viable artifacts and abstract theoretical constructs [41], the adoption of ADR in computer security research is still underdeveloped. Most notably, Chen et al. [37] adopted ADR to build a model to assess the risk of system downtime in computer networks due to the exploitation of software vulnerabilities. Puhakainen and Siponen [38] developed a theory-based IS security training approach, which they then validated empirically through an action research intervention. Other relevant applications of ADR include the work of Waguespack et al. [39] in the formulation of security design through Thriving Systems Theory, as well as the risk reporting prototype that Beer et al. [40] developed for information-dense IT projects.

Methodologically, any ADR project follows four well-defined stages: **(1)** Problem Formulation; **(2)** Building, Intervention, and Evaluation (BIE); **(3)** Reflection and Learning; **(4)** Formalization and Learning. The problem formulation of this research project is presented in the two preceding sections. In building, intervening and evaluating, we define the *alpha* version of the model (see Fig. 1). We then engage in reflection and learning to identify any new requirements that emerge from our exercise. Finally, we formalize our learning and prepare a second iteration of the model, which would lead to the development of the *beta* version of the model. This paper presents an early *alpha* version of our insider cyber-security threat model and identifies possible research directions in further refining the model towards the development of the *beta* version.

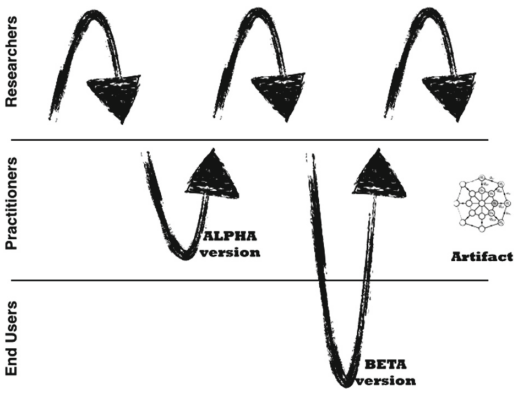


Fig. 1. Action Design Research adopted in this project (based on [7])

The development of an artifact is central to ADR. In our research, we consider artifacts as *ensembles*. As Orlikowski and Iacono [8] note, there are multiple ways in which IT artifacts come into being. In this research project, we articulate ADR following the 'technology as development process' conception of the artifact [8]. This view of the artifact focuses on the social processes that are related to the design, development and implementation of the technical artifact.

In our research, the artifact in question is the computer-based model for insider cyber-security threats. Using the Orlikowski and Iacono [8] vocabulary, this kind of artifact "examines the role of key stakeholders… [and] how such roles engender conflict, power moves, and symbolic acts." Our computer-based model offers valuable insights into the complex organizational and socio-political processes that are involved in the formulation and implementation of cyber-security strategies and policies. Hence, the model of insider threats becomes a management tool for constant monitoring and assessment of threats as these emerge. In due course, the artifact may reshape itself to influence structure of operations and the related business processes (as postulated by [9] among others). Finally, we argue that given an instrument that faithfully measures the constructs of the model as they manifest in the subjects, the responses of individuals are patterned with (and are therefore not independent of) organizational cyber-security rules, best practices, and technical controls. In other words, our instantiation of the human activity model is sensitive to cyber-security rules, best practices, and technical controls.

4 Artifact Description

To develop our insider threat model, we draw on a conceptual framework that is based on Hall's theory of *silent messages*. Hall [34] argues that silent messages are emanated via human activity interactions, such as language, greeting protocol, gender and status roles, and these get manifested at three levels – *formal*, *informal* and *technical*. In this paper, we argue that an understanding of these cultural streams at the *formal*, *informal* and *technical* levels can be critical for the defense of the organization against insider threats, and should, therefore, be assessed on a regular basis. In our human activity model for insider threats, there are two concepts that need elucidation. First, the Primary Message System (PMS) and cultural streams as proposed by Hall [34]. Second, the manifestation of these cultural streams in terms of *formal*, *informal* and *technical*. These are discussed below.

Hall's [34] taxonomy of behavioral patterns allows us to capture, at a moment in time, the attitudes of employees on matters of security. Each application of the taxonomy generates a panel of data allowing for analyses of changes in organizational security culture over time and between events of interest. In this paper we provide an example of how our adaptation of Hall's [34] taxonomy of behavioral patterns allows us to monitor and interpret the change in attitudes of different employees before and after the occurrence of a security breach. In his intercultural communication theory, Hall [34] argues that "no culture has devised a means for talking without highlighting some things at the expense of other things." Put another way, cultures demonstrate their idiosyncrasies in observable ways. He attempts to unearth different aspects of human behavior that are taken for granted and are often ignored by classifying them into streams that interact with each other – the silent messages. Laying emphasis on the concepts of time and space, Hall identified ten cultural streams that are rooted in physiology and biology. These streams form the primary message system (PMS), which is essentially a map of a wide variety of cultural and behavioral manifestations. A brief summary of each of the cultural streams is presented below:

- **Interactional (Int):** Speech is one of the most elaborate forms of interaction, and it is reinforced by the tone, voice and gesture. Interaction lies at the hub of the "universe of culture", and hence, every social action grows from it.
- **Organizational (Org):** Refers to the complexity of associations. Hall uses the analogy of bodies of complex organisms as being societies of cells. In his view, association begins when two cells join.
- **Economic (Eco):** Subsistence relates to the physical livelihood, income and work.
- **Gender (Gen):** Refers to the cultural differentiation of sexes and their interactions and associations.
- **Territoriality (Ter):** Refers to the division of space in everyday life, and ownership.
- **Temporality (Tem):** Refers to the division of time and sequencing.
- **Instructional (Ins):** Refers to a basic activity of life that includes learning and teaching. According to Hall, culture is shared behavior that is acquired, and not taught.
- **Recreational (Rec):** Refers to fun and recreation as part of work and occupation. However, this stream also captures competitive human behavior.
- **Protective (Pro):** Defense is a key element of any culture. Different cultures have developed different defense principles and mechanisms. The Protective stream is closely related to the Recreational stream since humor is often used as an instrument to hide or protect vulnerabilities.
- **Exploitational (Exp)**: Hall draws an analogy with the living systems to highlight the importance of evolutionary adaptation to "specialized environment conditions." By extension, the Exploitational stream represents the sophistication in the use of tools, techniques, materials and skills in a competitive and changing environment.

Hall [34] also argues that culture operates at three distinct levels, which are in a state of constant interaction: *formal*, *informal* and *technical*. His theory of culture is, essentially, a theory of change. Formal beliefs shape informal adaptations in behavior; and these, in turn, lead to technical analysis. Each of the ten streams is further broken down into *formal*, *informal* and *technical* to represent the different modes of behavior.

For each impact of the cultural stream at the three levels, we developed an emergent scenario. Each scenario took the form of a statement with a 5-point Likert scale to assess how members of the organization feel about the given situation (refer to Table 1).

5 Evaluation

In our research, we have employed our artifact to undertake an assessment of insider threats at a small, fledgling business. Crown Confections offers an assortment of confections for online or phone-in purchase. Customers have the option of phoning-in their orders or purchasing their pastries from Crown's website. Crown accepts credit card payments and uses PayPal's commercial suite to process these payments in a secure manner. The Crown staff consists of 20 individuals. The company wanted to develop a mechanism for monitoring insider threats. As researchers, we proposed the

Table 1. Framework depicting human activity interactions

	Int	Org	Eco	Gen	Ter	Tem	Ins
Formal	The tone used in communications affects my understanding of information security	My superiors provide comprehensive rules on security issues	I feel that I get the right compensation for the quality and dedication I put in my work	I consider a formal explication of gender issues to be important with respect to cyber security	I welcome suggestions from security professionals which may change the way my Department or work group operates	I prioritize security related tasks over other day-to-day tasks	I value the mentorship of senior colleagues regarding security and its interaction with my daily tasks
Informal	When faced with security warnings (for example a notice from an antivirus program that a download poses security risks), I always alter my behavior accordingly	My superiors casually stress the significance of rules on security issues	Money is my primary motivator with respect to work	I respect gender issues when dealing with cyber security issues	I welcome suggestions from security professionals, which may change the way I do my job	I actively follow my own schedule for performing security related tasks, such as changing my password, scanning my machine for viruses etc	I learn a lot about security issues by observing my peers
Technical	I understand the jargon used in information security communications	My superiors technically explain security rules and issues	The decision to undertake work related activities is a function of how much I get paid	I believe that the appearance of men and women working in technical aspects of security are consistent with their proficiencies	I am comfortable with discrepancies between actual and formal job descriptions, which have implications for security	I appreciate when security related tasks are being set out and enforced on a predetermined periodic basis	I understand the importance and relevance of information security training curricula

use of silent message theory to eventually develop an artifact in the form of a dashboard. It was agreed that the researchers would work with the Crown Confections employees to first develop an *alpha* version, which would then lead to the development of a *beta* version following evaluation and feedback from the end users. In this paper we are sharing the model as it has been shaped in the *alpha* version [7].

For the purposes of our research, we administered the human activity model presented here to the Crown Confections staff twice– once prior to the breach, and again after the breach. The breach included a spear-phishing and a brute force attack that compromised admin and user credentials. All members of the company provided usable responses. Our aim in administering the instrument at Crown Confections is to illustrate its ability to capture nuanced cultural aspects of an organization with respect to the attitudes, behaviors, and perceptions of its members concerning cyber-security.

Given the formative evaluation of the artifact during the *alpha*-cycle, we conducted multiple rounds of semi-formal interviews with organizational members to give them

the opportunity to share their experience and shape different aspects of the instrument. This feedback was invaluable as it helped us review key features of the instrument, such as the content and phrasing of certain questions. Alongside areas of improvement, organizational members praised the fact that the questions comprising the instrument touch upon aspects of cyber-security that are often overlooked in typical questionnaires of cyber-security risk assessment. For instance, organizational members highlighted the importance of examining issues of gender or the financial motivation of employees with respect to cyber-security. Furthermore, organizational members also agreed that regular monitoring of the instrument could pinpoint actionable insights on how to improve different aspects of cyber-security across the organization– which is our main research goal in this project. Hence, considering the application of the instrument in Crown Confections, it appears that the *alpha* version represents an insightful canvas for the examination of cyber-security as a diverse and granular socio-technical concept in contemporary organizations.

6 Discussion

The results of our artifact can provide useful visualizations for managers and executives who are interested in 'soft' aspects of cyber-security. Table 2 is a heat map that represents graphically the spectrum of positive and negative changes that were recorded in the post-breach perception of employees in relation to cyber-security. Positive changes indicate a move that is directed towards the high end of our Likert scale ('strongly agree'), whereas negative changes indicate a move in the opposite direction.

Table 2. Heat map of human activity interactions

	Int	Org	Eco	Gen	Ter	Tem	Ins	Rec	Pro	Exp
F	0.85	-0.5	-0.3	-0.4	1.45	1.45	-0.25	-0.75	-0.55	1.4
I	0.95	-1.55	0.15	0.05	0	0	1.25	-0.35	0.75	0.25
T	-0.7	-0.4	0.05	0.4	-0.65	0.75	1.4	-0.15	1	0.4

The different shades of green and white account for the different levels of gradation between the most positive and most negative changes. In Table 2, the positive changes are depicted with dark color accents, whereas the negative changes are depicted with white color accents. Shades in between indicate that no change or minor changes (positive or negative).

In terms of formalizing our learning, the analysis from the heat map can be represented graphically in the form of a radar map (see Fig. 2). The graphic representation shows how silent messages and attitudes of individuals change following a breach. Even without the occurrence of a breach, the radar maps can form the basis of an

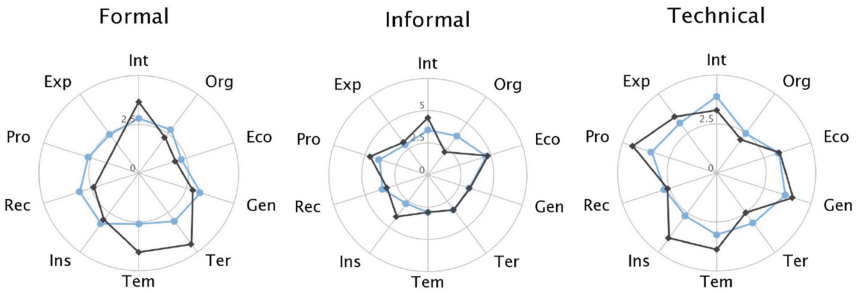


Fig. 2. Radar maps showing pre and post breach scenarios (blue line represents pre-breach situation and black line represents post breach) (Color figure online).

insider threat analytics dashboard that benchmarks and monitors different aspects of cyber-security culture. In this way, the model can help us build the cyber-security culture profile of a given organization.

In the following subsection, we will discuss a small number (three) of cases where a change in the perception of cyber-security has been recorded in our study.

6.1 Formal Layer

At the *formal* layer, we noticed a positive change in the Territorial stream, which examines how employees welcome suggestions from security professionals, which may change the way their Department or work group operates. A positive change shows higher agreement rates with the statement that comprises this cell, and it is, in fact, a reasonable reaction to a security breach. Following the breach, there may be an increasing need for openness as organizational members seek to share better security practices and get answers from others with more knowledge. However, an opposite reaction, namely a decreased rate of agreement with the statement, could also be seen as an expected outcome. The insider breach could cause organizational members to isolate themselves in small clusters that do not communicate with each other, or prevent them from reaching out to colleagues to openly discuss cyber-security related issues altogether. Similar effects have been noted in the literature with regards to the moral disengagement that security-related stress seems to produce [43]. This type of organizational stress triggers an "emotion-focused coping response in the form of moral disengagement from [information security policy] ISP violations, which in turn increases one's susceptibility to this behavior" [43, p. 285]. Stress that stems from work overload, complexity and uncertainty in response to a security breach could activate said coping mechanisms and provide the basis for the rationalization of non-compliant security behavior.

6.2 Informal Layer

At the informal layer, the Instructional stream refers to how receptive employees are to learning from peers. As the security breach represents a moment of organizational

crisis, organizational members increasingly seek help from their peers in order to mitigate difficulties associated with navigating the new, equivocal organizational situation. This would be particularly true in the case where there is a lack of a structured and well-prepared security incident response [44].

6.3 Technical Layer

The Interactional stream of the technical layer refers to the understanding of technical jargon in cyber-security issues and communications. Following the breach, there seems to be less agreement with the reference statement of this cell, which is a plausible outcome. A negative change indicates that some employees may feel challenged with the technical jargon associated with a security breach. Despite the increasing investment in cyber-security training and education, organizational members that occupy non-technical posts are not required to use this technical jargon in their day-to-day work. This means that it may be more difficult for them to understand the technical terminology and the details of how a security breach came about.

7 Conclusions

We have examined here the efficacy of a human activity model in providing insights to security culture where this efficacy can, in a sense be characterized as the sensitivity of the human activity model to changes in security culture. We highlight this sensitivity by examining pre and post breach instantiations of the model and the ensuing changes. We do this for a single organization. Naturally, generalizability of these findings to other organizational settings becomes "problematic." We argue here that this taken-as-given conception of generalization is inappropriate in this case. We have applied a theory (i.e. Hall's [34] human activity model) in a setting that is different from the one in which it was developed. Our findings as discussed in Sect. 5 demonstrate that our application of the theory is valid in our setting. This work is an example of type TE generalization – generalizing from theory to description Lee and Baskerville [42]. Lee and Baskerville write of this form of generalization: "researchers are interested not only in pure or basic research – the development, testing and confirmation of theories – but also in the utility of their theories in the actual business settings of executives, managers, consultants and other practitioners." By applying the human activity model in this setting we demonstrate its ability to be generalized to different settings and provide evidence consistent with its theoretical assertions.

There has been a limited amount of research that focuses on pre- and post-breach attitudinal changes. Berezina [35] undertook a study of hotel guest perceptions following a breach and found changes in perceived service quality, satisfaction and likelihood of recommending a hotel. Similarly, Rosoff [36] conducted a behavioral experiment to explore individual response to cyber-crimes. The results suggest differences in messages that individuals exhibit following a breach. The Rosoff [36] study was an experimental design, which covered predefined scenarios. In one scenario, Rosoff [36] explored if attack characteristics and attack mode had any influence of

victim's behavior following a breach. In the second scenario, they assessed if the motivation of the attacker and the resolution of the breach had any impact on the behavior of the victim.

While studies such as those of Rosoff [36] and Berezina [35] do shed some light on the manner in which breaches manifest and how individual behavior changes or how victims respond, they fall short of providing an ongoing assessment of the cyber-security situation. Our research project addresses this gap in the literature by presenting a proof-of-concept demonstration of a human activity model for insider cyber-security threats that considers the perceptions of insiders prior and after a security breach.

Future research will be concerned with the development of a *beta* version of our model. A more mature version of the instrument could be administered across different organizations and industries to get a more informed understanding of the granularity of cyber-security as a socio-technical concept. Different research contexts and organizational settings may result in different iterations of the instrument. Although Crown Confections helped researchers gather sufficient feedback on the design and functionality of the instrument, a larger organization could provide a more challenging research setting with a larger number of responses, and perhaps more contradictory responses and feedback that would lead to more fine-detailed iterations of the model.

References

1. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **SE-13**, 222–232 (1987)
2. Audit Commission, Losing an empire, finding a role. HMSO, London (1989)
3. Audit Commission, *Survey of computer fraud & abuse*. The Audit Commission for Local Authorities and the National Health Service in England and Wales (1990)
4. Audit Commission, *Opportunity makes a thief. Analysis of computer abuse*. The Audit Commission for Local Authorities and the National Health Service in England and Wales (1994)
5. Office of Technology Assessment, *Information security and privacy in network environments*. US Government Publication (1994)
6. Bowen, B.M., et al.: Designing host and network sensors to mitigate the insider threat. IEEE Secur. Priv. **7**(6), 22–29 (2009)
7. Sein, M., et al.: Action design research. MIS Q. **35**(1), 37–56 (2011)
8. Orlikowski, W.J., Iacono, C.S.: Research commentary: Desperately seeking the "IT" in IT research—A call to theorizing the IT artifact. Inf. Syst. Res. **12**(2), 121–134 (2001)
9. Jones, M.R., Karsten, H.: Gidden's structuration theory and information systems research. MIS Q. **32**, 127–157 (2008)
10. Dhillon, G., Moores, S.: Computer Crimes: theorizing about the enemy within. Comput. Secur. **20**(8), 715–723 (2001)
11. Warkentin, M.E., Willison, R.: Behavioral and policy issues in information systems security: The insider threat. Eur. J. Inf. Syst. **18**(2), 101–105 (2009)
12. Cappelli, D.M., et al.: Common Sense Guide to Prevention and Detection of Insider Threat, 3rd Edition—Version 3.1 (2009)

13. Cummings, A., et al.: Insider Threat Study: Illicit Cyber Activity Involving Fraud in the U.S. Financial Services Sector (Technical Report CMU/SEI-2012-SR-004) (2012)
14. Brancik, K.: Insider Computer Fraud: An In-depth Framework for Detecting and Defending against Insider IT Attacks. Auerbach Publications, Boca Raton (2007)
15. Ponemon Institute, Risk of Insider Fraud: Second Annual Study (2013)
16. Hartel, P.H., Junger, M., Wieringa, R.J.: Cyber-crime Science = Crime Science + Information Security. Technical Report TR-CTIT-10-34, CTIT, University of Twente, Oct 2010. http://eprints.eemcs.utwente.nl/18500/
17. Spitzner, L.: Honeypots: Catching the insider threat. In: Proceedings of 19th Annual Computer Security Applications Conference, pp. 170–179. IEEE, Las Vegas, NV, USA (2003)
18. Chagarlamudi, M., Panda, B., Hu, Y.: Insider threat in database systems: Preventing malicious users' activities in databases. In: Sixth International Conference on Information Technology: New Generations, 2009. ITNG 2009, pp. 1616–1620. IEEE, Las Vegas, NV, USA (2009)
19. Boss, S.R., et al.: If someone is watching, I'll do what I'm asked: Mandatoriness, control, and information security. Eur. J. Inf. Syst. **18**(2), 151–164 (2009)
20. Bulgurcu, B., Cavusoglu, H., Benbasat, I.: Information security policy compliance: an empirical study of rationality-based beliefs and information security awareness. MIS Q. **34** (3), 523–548 (2010)
21. D'Arcy, J., Hovav, A., Galletta, D.: User awareness of security countermeasures and its impact on information systems misuse: A deterrence approach. Inf. Syst. Res. **20**(1), 79–98 (2009)
22. Herath, T., Rao, H.R.: Encouraging information security behaviors in organizations: Role of penalties, pressures and perceived effectiveness. Decis. Support Syst. **47**(2), 154–165 (2009)
23. Guo, K.H., et al.: Understanding nonmalicious security violations in the workplace: a composite behavior model. J. Manage. Inf. Syst. **28**(2), 203–236 (2011)
24. Reason, J.: Achieving a safe culture: theory and practice. Work Stress **12**(3), 293–306 (1998)
25. Reason, J.T.: The human contribution: unsafe acts, accidents and heroic recoveries, p. 295. Ashgate, Farnham (2008)
26. Greitzer, F.L., Frincke, D.A.: Combining traditional cyber-security audit data with psychosocial data: Towards predictive modeling for insider threat mitigation. In: Probst, C.W., et al. (eds.) Insider Threats in Cyber-security - Advances in Information Security, 49, pp. 85–113. Springer, Heidelberg (2010)
27. Hoyer, S., et al.: Fraud prediction and the human factor: an approach to include human behavior in an automated fraud audit. In: 45th Hawaii International Conference on System Sciences Proceedings (HICSS), Grand Wailea, pp. 2382–2391. IEEE Computer Society, Maui, HI, USA (2012)
28. Vance, A., Siponen, M., Pahnila, S.: Motivating IS security compliance: insights from habit and protection motivation theory. Inf. Manage. **49**(3), 190–198 (2012)
29. Herath, T., Rao, H.R.: Protection motivation and deterrence: A framework for security policy compliance in organisations. Eur. J. Inf. Syst. **18**(2), 106–125 (2009)
30. Ramachandran, S., et al.: Variations in information security cultures across professions: A qualitative study. Commun. Assoc. Inf. Syst. **33**, 163–204 (2013)
31. Hedström, K., et al.: Value conflicts for information security management. J. Strateg. Inf. Syst. **20**(4), 373–384 (2011)
32. Talib, Y.A., Dhillon, G.: Invited paper: Employee emancipation and protection of information. In: 5th Annual Symposium on Information Assurance (ASIA 2010) (2010)

33. Dhillon, G., Chowdhuri, R., Pedron, C.: Organizational transformation and information security culture: A telecom case study. In: Cuppens-Boulahia, N., Cuppens, F., Jajodia, S., Abou El Kalam, A., Sans, T. (eds.) SEC 2014. IFIP AICT, vol. 428, pp. 431–437. Springer, Heidelberg (2014)
34. Hall, E.T.: The silent language, 2nd edn. Anchor Books, New York (1959)
35. Berezina, K., et al.: The impact of information security breach on hotel guest perception of service quality, satisfaction, revisit intentions and word-of-mouth. Int. J. Contemp. Hospitality Manage. **24**(7), 991–1010 (2012)
36. Rosoff, H., Cui, J., John, R.: Behavioral experiments exploring victims' response to cyber-based financial fraud and identity theft scenario simulations. In: Tenth Symposium on Usable Privacy and Security (SOUPS), pp. 175–186. USENIX Association, Menlo Park, CA, USA (2014)
37. Chen, P.-Y., Kataria, G., Krishnan, R.: Correlated failures. Diversification, Inf. Secur. Risk Manage. MIS Q. **35**(2), 397–422 (2011)
38. Puhakainen, P., Siponen, M.: Improving employees' compliance through information systems security training: An action research study. MIS Q. **34**(4), 757–778 (2010)
39. Waguespack, L.J., Yates, D.J., Schiano, W.T.: Towards a design theory for trustworthy information systems. In: 47th Hawaii International Conference on System Sciences (HICSS), pp. 3707–3716 (2014)
40. Beer, M., Meier, M.C., Mosig, B., Probst, F.: A prototype for information-dense it project risk reporting: an action design research approach. In: 47th Hawaii International Conference on System Sciences (HICSS), pp. 3657–3666 (2014)
41. Gregor, S., Hevner, A.R.: Positioning and presenting design science research for maximum impact. MIS Q. **37**(2), 337–355 (2013)
42. Lee, A., Baskerville, R.L.: Generalizing generalizability in information systems research. Inf. Syst. Res. **14**(3), 221–243 (2003)
43. Markus, M.L.: Power, politics, and mis implementation. Commun. ACM **26**(6), 430–444 (1983)
44. Plester, B.: Execution of a joke: Types and functions of humour. In: The Complexity of Workplace Humour: Laughter, Jokers and the Dark Side of Humour, pp. 39–66. Springer, Heidelberg (2016)

Evaluating CVSS Base Score Using Vulnerability Rewards Programs

Awad Younis(✉), Yashwant K. Malaiya, and Indrajit Ray

Colorado State University, Fort Collins, CO 80523, USA
{younis,malaiya,indrajit}@CS.ColoState.EDU

Abstract. CVSS Base Score and the underlying metrics have been widely used. Recently there have been attempts to validate them. Some of the researchers have questioned the CVSS metrics based on a lack of correlation with the reported exploits and attacks. In this research, we use the independent scales used by the vulnerability reward programs (VRPs) to see if they correlate with the CVSS Base Score. We examine 1559 vulnerabilities of Mozilla Firefox and Google Chrome browsers. The results show that there is a significant correlation between the VRPs severity ratings and CVSS scores, when three level rankings are used. For both approaches, the sets of vulnerabilities identified as Critical or High severity vulnerabilities include a large number of shared vulnerabilities, again suggesting mutual conformation. The results suggest that the CVSS Base Score may be a useful metric for prioritizing vulnerabilities, and the notable lack of exploits for high severity vulnerabilities may be the result of prioritized fixing of vulnerabilities.

Keywords: CVSS Base score · Vulnerability reward program · Bug bounty programs · Software vulnerability severity · Vulnerability exploitation · Software defensive techniques · Exploit mitigation techniques

1 Introduction

Assessing the risk associated with individual software vulnerabilities is accomplished by assessing their severity. CVSS Base score is the de facto standard that is currently used to measure the severity of vulnerabilities [1]. Evaluating the accuracy of the CVSS Base score is very important as they are intended to help decision makers in resource allocation, patch prioritization, and risk assessment. The lack of evaluation makes CVSS usability risky and that could lead to a waste of limited resources available or a breach with a high impact.

Vulnerability exploitability can be affected by the existence of defense techniques. The main objective of the defense techniques is to reduce the likelihood that the efforts of attackers will succeed [2]. As 92 % of reported vulnerabilities are in software not in networks [3], recently vendors such as Microsoft, Cisco, Google, Mozilla, etc. have added new defensive layers at the software

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 62–75, 2016.
DOI: 10.1007/978-3-319-33630-5_5

level. Among them are secure development lifecycle, vulnerability mitigation techniques, sandbox, and VRPs. VRPs are programs adopted by software vendors to pay security researchers, ethical hackers and enthusiasts for exchange of discovering vulnerabilities in their software and responsibly disclosing the findings to the vendors [4]. Having more eyes on the code means that VRPs uncovered many more vulnerabilities and that makes finding vulnerabilities more difficult for malicious actors and hence ensure the security of software. Besides, vulnerabilities found by VRPs results in a coordinated disclosure and patch that minimizes the risk of vulnerabilities discovery and exploitation [5]. Our approach is inspired by the economics of exploitation model proposed by Miller et al. in [13]:

$$\begin{aligned} AttackerReturn = (Gain\ per\ use \times Opportunity\ to\ use) - \\ (Cost\ to\ acquire\ vulnerability + Cost\ to\ weaponize) \end{aligned}$$

The authors argue that an attacker must invest resources to acquire vulnerabilities and develop weaponized exploit for it. While mitigation techniques have shown to increase the cost and complexity of developing an exploit and hence cost of weaponize [14], we argue that VRPs can also be an important factor in this equation. As software vendors invest significantly to find vulnerabilities the cost of an attacker acquiring a vulnerability is going to be increased and that reduces the likelihood of vulnerabilities discovery and exploitation. Many software vendors such as Google, Mozilla, Facebook, PayPal, and recently Microsoft have adopted using VRPs. They realized that attackers are finding vulnerabilities faster and thus adapting VRPs will help put more sets of eyes looking for vulnerabilities and that makes all vulnerabilities shallow.

There are a number of VRPs and each one of them have their rules and criteria. Among these programs are Mozilla Firefox VRP [6] and Google Chrome VRP [7]. Mozilla Firefox and Google Chrome VRPs determine the reward amount of a vulnerability based on its severity and proof of its exploitation. Both VRPs classify the severity of vulnerabilities as critical, high, medium and low. The details about the description of every severity level for Firefox and Chrome VRPs can be found respectively in [8,9] respectively. While Firefox VRP rewards amount ranges from 500–10,000, Chrome VRP rewards ranges from 500–60000. Firefox VRP pays only for vulnerabilities that has been rated by VRP Committee as a critical or a high and some moderate vulnerabilities, while Chrome VRP rewards critical, high, medium, and some low vulnerabilities.

Problem Description. Recently, there have been efforts to validate CVSS Base score. Some of the researchers have evaluated CVSS Base score using reported exploits [10,11] and attacks [12]. The results show that CVSS Base score has a poor correlation with the reported exploits ([10,11]) and with the reported attacks [12]. Thus, CVSS Base scores have been considered not a good risk indicator [12] because the majority of vulnerabilities have high scores and have no reported exploits or attacks. Hence, it is hard to use those scores to prioritize among vulnerabilities. However, the lack of exploits or attacks may be a result of prioritized fixing of vulnerabilities.

Contribution. In this research, we propose using independent scales used by VRPs to evaluate CVSS Base score. VRPs use their own vulnerability severity rating systems that use a very thorough technical analysis and security experts opinions to assign a severity to vulnerabilities. The severity ratings are then used to pay money ranging from 500$ to 60,000$ or even more. Hence, comparing CVSS Base score with VRPs severity ratings could explain whether CVSS high scores are reasonable, and why having many high sever vulnerabilities with no reported exploit or attacks.

To conduct this study, we examine 1559 vulnerabilities of Mozilla Firefox and Google Chrome browsers. The two software has been selected because of their rewarding programs maturity and their rich history of publicly documented rewarded vulnerabilities. Besides, the examined vulnerabilities have been assessed by both VRPs rating systems and the CVSS Base score which makes their comparison feasible.

The paper is organized as follows. Section 2 presents the related work. In Sect. 3, the selected datasets are presented. In Sect. 4, the validity of CVSS Base score is examined. Section 5 presents the discussion. In Sect. 6, concluding comments are given and the issues that need further research are identified.

2 Related Work

Bozorgi et al. [10] have studied the exploitability metrics in CVSS Base metrics. They argued that the exploitability measures in CVSS Base metrics do not differentiate well between the exploited and not exploited vulnerabilities. They attributed that to the fact that many vulnerabilities with a CVSS high score have no reported know exploit and many vulnerabilities with low CVSS scores have a reported know exploit. However, in this paper, we evaluate the performance of CVSS Base score considering both the exploitability and the impact factors using vulnerability rewards programs and we provide an insight into why many vulnerabilities with a high CVSS score and have no exploits.

Allodi and Massacci in [12] have used a case-control study methodology to evaluate whether a high CVSS score or the existence of proof of concept exploit is a good indicator of risk. They use the attacks documented by Symantecs AttackSignature as the ground truth for the evaluation. Their results show that CVSS Base score performs no better than randomly picking vulnerabilities to fix. Besides, they also show that there are many vulnerabilities that have a high CVSS score and are not attacked. However, in this paper, we seek to find an explanation of why the majority of vulnerabilities have a high CVSS score and have no reported exploits. Thus, we use VRPs instead of an attack in the wild to conduct this study.

Younis and Malaiya in [11] have compared Microsoft rating systems with CVSS Base metrics using the availability of exploits as a ground truth for the evaluation. In addition to finding that both rating systems do not correlate very well with the availability of exploit, they also find that many vulnerabilities have a high CVSS score and have no reported exploits. However, in this study we try

to use different ground truth for the evaluation so that an explanation for why many vulnerabilities have a high CVSS scores and have no reported exploits may be provided.

Finifter et al. in [4] have examined the characteristics of Google Chrome and Mozilla Firefox VRPs. The authors find that using VRPs helps improving the likelihood of finding latent vulnerabilities. They also find that monetary rewards encourage security researchers not to sell their result to the underground economy. Besides, they find that patching vulnerabilities found by the VRPs increases the difficulties and thus the cost for the malicious actors to find zero-day vulnerabilities or exploit them. However, in this study, we examine using VRPs as ground truth to evaluate CVSS Base score.

Swamy et al. in [14] at the Microsoft Security Response Center examine the impact of using exploit mitigation techniques that Microsoft has implemented to address software vulnerabilities. One of their result shows that stack corruption vulnerabilities that were historically the most commonly exploited vulnerability class are now rarely exploited. However, in this research, we focus on the impact of using VRPs on the availability of exploits and on the relationship between VRPs measures and CVSS Base score.

3 Datasets

In this section, we first provide the source of the data. Then we show how the data were collected and analyzed. In this research, the data about vulnerabilities, exploits, and vulnerabilities rewards program data have been collected from the National Vulnerability Database (NVD) [15], Exploit Database (EDB) [16], and Mozilla Firefox [17] and Google Chrome bug databases [18] receptively. Table 1 shows the number of the examined vulnerabilities and their exploits. It should be noted that the total number of the Firefox vulnerabilities is 742. Out of this number, a 195 vulnerabilities were not examined because we could not find information about them and that is explained as follows. First, 71 vulnerabilities have no direct mapping between the Common Vulnerabilities and Exposures (CVE) number and the Firefox Bug ID. Second, a 122 vulnerabilities could not be accessed due to the unauthorized access permission (You are not authorized to access this data); Third, two vulnerabilities have no data recorded in the Firefox bug database. We found that the VRP data in the Firefox bug database have started to be recorded starting 2009. Thus, all vulnerabilities and exploits of Firefox during the period 2009 to October 2015 were collected. On

Table 1. Firefox and Chrome vulnerabilities

Software	Vulnerabilities	Exploit exist
Firfox	547	22
Google Chrome	1012	5

the other hand, it should also be noted that the total number of vulnerabilities of Chrome is 1084. A 72 vulnerabilities were not examined because we could not find information about them because of the unauthorized access permission "You are not authorized to access this data". We also found that the VRP data in the Chrome bug database have started to be recorded starting 2010. Therefore, all vulnerabilities and exploits of Chrome during the period 2010 to October 2015 were collected. The data of every examined vulnerability of Firefox and Chrome were collected using the following steps. First, from NVD, the vulnerability is first identified. Next, for every existing link in NVD to vendors' bug database, we collected the vulnerability's severity rating and rewards data assigned by the VRPs. After that, for every vulnerability's CVE number found in the vendors bug database, the CVSS scores and severity values were collected. Lastly, for every examined vulnerability we used the CVE number to verify whether it has an exploit reported in the EDB or not.

3.1 Firefox Vulnerabilities Analysis

Table 2 shows only three of the Firefox vulnerabilities because showing the whole vulnerabilities is limited by the number of pages allowed. Firefox VRP does not provide data about the amount of the reward paid and rather it uses: (1) + symbol to indicate the bug has been accepted and payment will be made, (2) - symbol to indicate the bug does not meet the criteria and payment will not be paid, and (3) ? symbol to indicate the bug is nominated for review by the bounty committee [8].

Table 2. The obtained measures of Firefox and CVSS Base score

CVE	Mozilla Firefox VRP CVSS Base score				Exploit existence
	Reward	VRP severity	Severity	Score	
CVE-2011-2371	3000-7500	sec-critical	High	10	EE
CVE-2013-1727	500-2500	sec-moderate	Medium	4	NEE
CVE-2015-0833	3000-5000	sec-high	Medium	6.9	NEE

The CVSS Base score assigns a score in the range [0.0, 10.0]. This score represents the intrinsic and fundamental characteristic of a vulnerability and thus the score does not change over time. CVSS score from 0.0 to 3.9 corresponds to Low severity, 4.0 to 6.9 to Medium severity and 7.0 to 10.0 to High severity. Mozillas security ratings are see-critical: vulnerabilities allow arbitrary code execution, sec-high: vulnerabilities allow obtain confidential data, sec-moderate: vulnerabilities which can provide an attacker additional information, sec-low: minor security vulnerabilities such as leaks or spoofs of non-sensitive information. We have found that 13 vulnerabilities did not meet the criteria for rewarding and

hence have been assigned "–" symbol. We have also found that 11 of them have a low and a moderate severity (five are low and six are moderate) and two are high and critical.

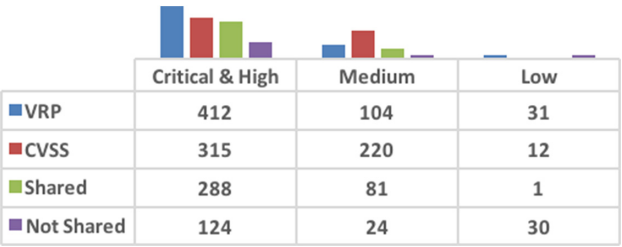
Table 3 shows the number of the rewarded and not rewarded vulnerabilities and their severity values for Firefox dataset. It should be noted that the Not Rewarded vulnerabilities are most likely have been discovered by internal discoverers (41.13 %) whereas Rewarded vulnerabilities have been discovered by external discoverers (58.87 %) [4]. While the Firefox bug database does not clearly provide information about whether the vulnerabilities have been discovered internally or externally, this was very clear in the Chrome bug database where the name and the team the discoverer works with is provided. Table 3 also shows the severity values and their frequency for rewarded and on rewarded data. It should be noted that the majority of the medium severity vulnerabilities and all low severity vulnerabilities were discovered internally.

Table 3. Firefox dataset

Vulnerabilities	Rewarded	Not rewarded
547	225	322
VRP severity	Rewarded	Not rewarded
Critical & High	210	202
Medium	15	89
Low	0	31

Figure 1 shows the vulnerabilities severity values of CVSS Base score and Firefox VRP ratings of Firefox dataset. There are 412 vulnerabilities that have been assessed as critical or high severity by VRP rating system whereas there are 312 vulnerabilities that have been assessed as high severity by CVSS Base score. It should be notated that Firefox VRP severity rating is the baseline that we are comparing CVSS scores with. It should also be noted that Shared means the same vulnerabilities, which have the same CVE number, that have been assigned the same severity value by Firefox VRP rating system and CVSS Base score. On the other hand, Not Shared means the same vulnerabilities, but have been assigned different severity values by CVSS Base score. Almost 70 % (69.9) of the vulnerabilities that have been assessed by Firefox VRP as critical or high severity have also been assessed as high severity by CVSS. Using Common Weakness Exposure (CWE) [19], which is used to identify vulnerabilities types, we have found that the majority of the Shared vulnerabilities are of the vulnerabilities that execute code. However, the 124 vulnerabilities that are Not Shared have all been assigned a high or critical severity by VRPs rating system, whereas CVSS Base score has assigned to 7 of them a low severity and to 117 of them a moderate severity.

On the other hand, almost 78 % (77.88) of the Shared vulnerabilities that have been assessed by Firefox VRP as a medium severity have also been assessed



	Critical & High	Medium	Low
VRP	412	104	31
CVSS	315	220	12
Shared	288	81	1
Not Shared	124	24	30

Fig. 1. Comparing Firefox VRP and CVSS severity values

as a medium severity by CVSS. However, there are 24 Not Shared vulnerabilities that have been assigned a medium severity by the VRP rating system. Out of these, 19 vulnerabilities have been assigned a high severity and five have been assigned a low severity. However, only one vulnerability that has been assessed as a low severity by CVSS base score and VRP rating system. While 30 vulnerabilities that have been assessed as a low severity by the VRP rating system, eight have been assessed as a high severity and 22 as a medium severity.

Table 4 shows the vulnerabilities that have been mismatched by CVSS Base score. As can be seen, seven vulnerabilities have been assessed as low severity by CVSS whereas three of them have been assessed as critical and four of them has been assessed as high severity by the VRP. It has noticed that those seven vulnerabilities have been assigned critical and high severity values by the Firefox VRP during the debate time, but the vulnerabilities severity first assignments were later changed [20]. We have found that the majority of those vulnerabilities requires unusual users interactions. We have also noticed that CVSS version 3, which has not been used yet, have consider using user interaction factor when assessing exploitability factor [21]. It is clear that the medium range of CVSS scores makes the main part of the mismatch compared to the high and low ranges.

Table 4. Vulnerabilities mismatched by CVSS Base score

VRP	Crtical		High		Moderate	Low	
CVSS	Low	Medium	Low	Medium	Low	Medium	High
Total	3	**24**	4	**93**	4	**22**	8

3.2 Chrome Vulnerabilities Analysis

The Chrome vulnerabilities have been examined similar to the Firefox vulnerabilities as shown in Table 3. The only difference is that Chrome bug database provides the amount rewarded. Chrome security ratings are similar to that of

Table 5. Chrome dataset

Vulnerabilities	Rewarded	Not rewarded
1012	584	428
VRP severity	Rewarded	Not rewarded
Critical & High	441	175
Medium	136	137
Low	7	116

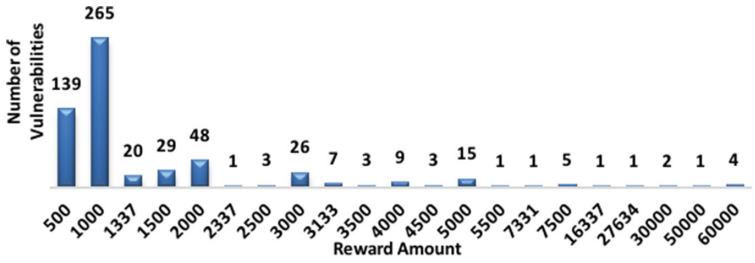
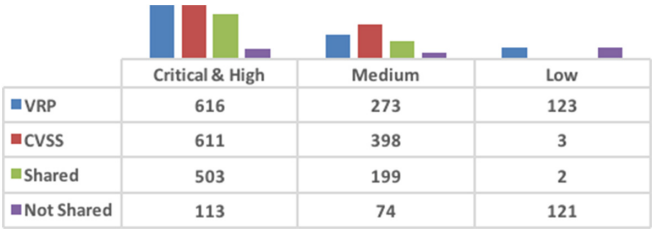


Fig. 2. Rewarded amount of Chrome rewarded vulnerabilities

Mozilla. Unlike Firefox where low severity vulnerabilities are not rewarded, seven low severity vulnerabilities have been rewarded by Chrome VRP. The seven low severity vulnerabilities have been found to effect non-critical browser features, crash inside the sandbox, or hang the browser.

Table 5 shows the number of the rewarded and not rewarded vulnerabilities and their severity values for Chrome dataset. We have found nine vulnerabilities have been classified as TBD (To Be Determined) and thus considered them as not rewarded. It should be noted that the majority of the Not Rewarded vulnerabilities have been discovered by Google internal discoverers and they represent around 41.4 %, whereas the Rewarded vulnerabilities have been discovered by external discoverers and represents around 57.7 %. Table 5 also shows the severity values and their frequency for the rewarded and not rewarded data. It should be noted that while the majority of the critical and high vulnerabilities have been discovered externally, the majority of the low vulnerabilities have been discovered internally. The frequency of the amount paid is shown in Fig. 2. As can be seen, the majority of the rewarded vulnerabilities (404) have been paid either 500$ or 1000$. We have noticed that 70.79 % (286) of those vulnerabilities have been assigned a high severity, 27.47 % (111) have been assigned a medium severity, and only 1.73 % (7) have been assigned a low severity by VRP. Looking at the point number 3–5 under the Reward amounts section in [7], we can see that establishing exploitability or providing a Proof of Concept (PoC) or with a poor quality of PoC could be the reason for paying less for many severe vulnerabilities.



	Critical & High	Medium	Low
VRP	616	273	123
CVSS	611	398	3
Shared	503	199	2
Not Shared	113	74	121

Fig. 3. Comparing Chrome VRP and CVSS severity values

Figure 3 shows the vulnerabilities severity of CVSS Base score and Chrome VRP rating system of Chrome dataset. Almost 82 % (81.65) of the vulnerabilities that have been assessed by Chrome VRP as critical or high severity have also been assessed as high severity by CVSS. However, the 113 vulnerabilities that are Not Shared have all been assigned a high severity by CVSS, whereas VRPs rating system have assigned to 35 of them a low severity and to 78 of them a medium severity. On the other hand, almost 73 % (72.89) of the Shared vulnerabilities that have been assessed by Chrome VRP as a medium severity have also been assessed as a medium severity by CVSS. However, there are 74 Not Shared vulnerabilities that have been assigned a medium severity by CVSS. Out of these, 73 vulnerabilities have been assigned a high severity and only one has been assigned a low severity. However, only two vulnerabilities that have been assessed as a low severity by CVSS Base score and VRP rating system, whereas out of the 121 that have been assigned a low severity by VRP, 35 have been assigned a high severity and 86 have been assigned medium severity by CVSS Base score. Table 6 shows the vulnerabilities that have been mismatched by CVSS Base score. We have noticed that out of the 113 vulnerabilities, 75 vulnerabilities have been assigned a high medium score 6.8.

Table 6. Vulnerabilities mismatched by CVSS Base score

VRP	Crtical		High		Moderate	Low	
CVSS	Low	Medium	Low	Medium	Low	Medium	High
Total	0	**1**	0	**112**	1	**86**	35

4 Validation of CVSS Base Score

In this section, we compare CVSS Base score severity with VRPs severity ratings. We assume that VRPs severity rating values are the ground truth because of the through technical analysis and security experts opinions used and the fact that the severity rating are used to pay money. To evaluate the performance of CVSS Base score, we describe when a condition (true or false) is positive or negative as follows:

True Positive (TP)	When the CVSS Base score assigns a high severity and VRPs assign critical or high, or, When CVSS Base assigns medium and VRPs assign medium
True Negative (TN)	When the CVSS Base score assigns low severity and VRPs assign low
False Negative (FN)	When the CVSS Base score assigns low and VRPs assign critical or high, or, When the CVSS Base score assigns medium and VRPs assign critical or high, or, When the CVSS Base score assigns low and VRPs assign medium
False Positive (FP)	When the CVSS Base score assigns and VRPs assign medium or low. Or when the CVSS Base score assign medium and VRPs assign low

Since CVSS Base score uses an ordinal range: 0–3.9 = Low, 4–6.9 = Medium, and 7-10 = High, the possibility of overlapping between the ranges could make high Low vulnerability such as 3.9 close to medium and high Medium such as 6.9 close to high. To take this into consideration, we used a cluster algorithm to group severity ranges based on the distance between their values. We implemented the K-Means clustering algorithm provided by R language [23] to cluster CVSS Base score for Firefox and Chrome dataset. The result for Firefox vulnerabilities show that Low is in the range from 1.9–5.4, Medium from 5.8–7.6, and High from 8.3–10, whereas the results for Chrome vulnerabilities show that Low is in the range from 2.6–5.1, Medium from 5.8–7.1, and High from 7.5–10.

We used statistical measures, termed sensitivity, precision, and F-measure to evaluate the performance of CVSS Base score severity. Sensitivity, which also termed recall, is defined as the ratio of the number of vulnerabilities correctly assessed as high or medium to the number of vulnerabilities that are actually high or medium as shown by the following: Sensitivity = TP/TP + FN. Precision, which is also known as the correctness, is defined as the ratio of the number of vulnerabilities correctly assessed as high or medium to the total number of vulnerabilities assessed as high or medium as shown by the following: Precision = TP/TP + FP. For convenient interpretation, we express these two measures in terms of percentage, where a 100 % is the best value and 0 % is the worst value. Both precision and sensitivity should be as close to the value 100 as possible (no false positives and no false negatives). However, such ideal values are difficult to obtain because sensitivity and precision often change in opposite directions. Therefore, a measure that combines sensitivity and precision in a single measure is needed. F-measure can be interpreted as the weighted average of sensitivity and precision. It measures the effectiveness of a prediction with respect to a user attached β times as much importance to sensitivity as precision. The general formula for the F-measure is shown by the following:

$$F_\beta - Measure = \frac{(1+\beta^2) \times Precision \times Senetivity}{(\beta^2 \times Precision) + Senetivity}$$

β is a parameter that controls a balance between sensitivity and precision. When $\beta = 1$, F-measure becomes to be equivalent to the harmonic mean, whereas when $\beta < 1$ it becomes more precision oriented. However, when $\beta > 1$, F-measure becomes more sensitivity oriented. In this paper β has been chosen to be 2. Due to their importance, we have also used the FP rate measure: FP rate = FP/FP + TN and the FN rate measure: FN rate = FN/TP + FN.

4.1 Result

To calculate the above mentioned performance measures, we need to obtain the confusion matrix for the two datasets. Using the severity ratings assigned by VRPs and CVSS Base score, the confusion matrix was determined as shown in Table 7. We have also determined the CVSS Base score ranges obtained by the clustering algorithm and due to the limited pages allowed we only show the results for the CVSS Base score original ranges. It should be noted that we add up the number of every condition, for instance True Positive for Fire fox = 369. As can be seen, CVSS scores before the clustering have a very high FP rate. Using the values in Table 7, the performance measures for CVSS Base score original ranges, clustering ranges and our mismatching analysis ranges have been calculated as shown in Table 8. We also used Spearman correlation measure to assess the correlation between CVSS scores before clustering and after clustering as shown in Table 9. As can be seen, CVSS score correlate with VRPs rating values with p-value less than 0.0001. Clustering score and using mismatching analysis have shown a slight improvement on the correlation value for the Chrome vulnerabilities whereas no effect have been noticed on the correlation value for Firefox vulnerabilities. However, we also looked at the percentage of the vulnerabilities that have been assigned high and medium severity by CVSS scores and VRPS ratings to verify which measure is more aggressive. For the whole dataset, VRPs

Table 7. CVSS Base score compared to VRPs rating systems

Condition	CVSS Vs. Actual VRPs	Firefox	Chrome
True Positive	When the CVSS High and VRPs Critical or High	288	503
	When CVSS Medium and VRPs Medium	81	199
True Negative	When CVSS Low and VRPs Low	1	2
False Negative	When CVSS Low and VRPs Critical or High	7	0
	When CVSS Says Medium and VRPs Critical or High	117	113
	When CVSS Low and VRPs Medium	4	1
False Positive	When CVSS High and VRPs Low or Medium	27	108
	When CVSS M and VRPs Low	22	86

Table 8. Performance measures for CVSS before and after clustering

Software	Performance measures	CVSS Scores before clustering (%)	CVSS Scores after clustering (%)
Firefox	Sensitivity	74.25	56
	Precision	88.25	93
	F1-Measure	80.66	70.21
	F2-Measure	57.99	46.94
	False Positive Rate	98	50
	False Negative Rate	25.75	43.54
Chrome	Sensitivity	86	66
	Precision	78	82
	F1-Measure	82	73
	F2-Measure	63	52
	False Positive Rate	99	60
	False Negative Rate	14	34

Table 9. Spearman correlation between CVSS Base score and VRPs rating system

Software	Correlation	CVSS Scores before clustering	CVSS Scores after clustering
Firefox	Value	0.65	0.47
	P-value	0.0001	0.0001
Chrome	Value	0.53	0.59
	P-value	0.0001	0.0001

have assigned 66 % of the vulnerabilities a high severity, whereas CVSS have assigned 59 % of the vulnerabilities a high severity. On the other hand, VRPs have assigned 24 % of the vulnerabilities a medium severity, whereas CVSS have assigned 46 % a medium severity.

4.2 Threats to Validity

In this research, we have considered two datasets of two software of the same domain, internet browsers. We consider extending our analysis as long as the data about software from different domains are publicly available and accessible. We are also aware that there are other factors that can affect the vulnerabilities exploitation. Thus, we in no way imply that VRPs should be the only consideration when trying to assess CVSS Base score.

5 Discussion

Results have shown that CVSS scores have a higher FP rate. This is mainly because of the number of True Negatives. Out of the 131 vulnerabilities that have been assigned as Low by chrome VRP only two (True Negative) vulnerabilities

have been assessed as Low by CVSS. We have found that 86 of these vulnerabilities have been assigned medium and 35 have been assigned high. On the other hand, out of the 31 vulnerabilities that have been assessed as Low by Firefox VRP, only one (True Negative) vulnerabilities have assessed as Low by CVSS. We have found that 22 of these vulnerabilities have been assigned medium and 8 have been assigned high.

There are more Execute Code vulnerabilities in Firefox than in Chrome. This could be explained by the effect of defensive mechanism, Sandbox, used by Chrome. Furthermore, based on the amount paid, the data from Chrome show that proving exploitability is more valuable than discovering vulnerabilities.

6 Conclusion and Future Work

This study evaluates CVSS Base Scores as a prioritization metric by comparing it with VRP reward levels, which are arguably more direct measures. We used 1559 vulnerabilities from Mozilla Firefox and Google Chrome browsers to conduct this study. The performance measures and the correlation results show that CVSS Base Score is suitable for prioritization. The fact that there are more vulnerabilities with a high CVSS scores and have no exploits or attacks have been explained by the effect of VRPs on vulnerabilities exploitation. Besides, considering that CVSS score assess most of the vulnerabilities as severer, data show that VRPs have assessed even more vulnerabilities as severe more than CVSS Base score.

Still, there appears to be a need for continued updating of the CVSS metrics and measures. CVSS should highly consider including the Likelihood of Exploit factor (not only the availability of exploit, but also how likely it is that functioning exploit code will be developed) as CWSS [24] and Microsoft [25] rating systems did. Besides. The two chosen VRPs rating systems have shown that Likelihood of Exploit is the main factor that determine the amount of the reward paid for the discoverers and that was very evident in Chrome dataset. As the two datasets considered represent two software of the same domain, examining data from different domains can be valuable as long as their data is publicly available and accessible.

References

1. Mell, P., Scarfone, K., Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0, p. 123. Published by FIRST-Forum of Incident Response and Security Teams (2007)
2. Defense in Depth. http://www.nsa.gov/ia/_files/support/defenseindepth.pdf. Accessed on 08 January 2016
3. Pescatore, J.: Application Security: Tools for Getting Management Support and Funding. White Paper, SANS Institute (2013)
4. Finifter, M., Devdatta, A., David, W.: An empirical study of vulnerability rewards programs. In: Proceedings of the 22nd USENIX Security Symposium, Washington, pp. 273–288 (2013)

5. Reading, D.: Connecting The Information Security Community. http://www.darkreading.com/coordinated-disclosure-bug-bounties-help-speed-patches/d/d-id/1139551
6. The Mozilla Security Bug Bounty Program. https://www.mozilla.org/en-US/security/bug-bounty/. Accessed on 08 January 2016
7. Chrome Reward Program Rules. https://www.google.com/about/appsecurity/chrome-rewards/index.html. Accessed on 08 January 2016
8. Security Severity Ratings. https://wiki.mozilla.org
9. Severity Guidelines for Security Issues. https://www.chromium.org/developers/severity-guidelines. Accessed on 08 January 2016
10. Younis, A.A., Malaiya, Y.K.: Comparing and evaluating CVSS base metrics and microsoft rating system. In: The 2015 IEEE International Conference on Software Quality, Reliability and Security, Vancouver, BC, pp. 252–261 (2015)
11. Allodi, L., Massacci, F.: Comparing vulnerability severity and exploits using case-control studies. J. Tra. Info. Syst. Secu. **17**(1), 1–20 (2014)
12. Miller, M., Burrell, T., Howard, M.: Mitigating Software Vulnerabilities. Technical report, Microsoft Security Engineering Center (2011)
13. Nagaraju, S.S., Craioveanu, G., Florio, E.: Software Vulnerability Exploitation Trends. Technical Report, Microsoft Trustworthy Computing Security (2013)
14. Bozorgi, M., Saul, L.K., Savage, S., Voelker, G.M.: Beyond heuristics: learning to classify vulnerabilities and predict exploits. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, pp. 105–114 (2010)
15. National Vulnerability Database. https://nvd.nist.gov/. Accessed on 08 January 2016
16. Exploit Database. https://www.exploit-db.com/. Accessed on 08 January 2016
17. Security Advisories for Firefox. https://www.mozilla.org/en-US/security/known-vulnerabilities/firefox/. Accessed on 08 January 2016
18. Chromium. https://code.google.com/p/chromium/issues/list. Accessed on 08 January 2016
19. Common Weakness Enumeration (CWE). http://cwe.mitre.org/. Accessed on 08 January 2016
20. Mozilla Bugzilla. https://bugzilla.mozilla.org/. Accessed on 08 January 2016
21. Common Vulnerability Scoring System, V3 Development Update. https://www.first.org/cvss. Accessed on 08 January 2016
22. Point-Biserial. https://www.andrews.edu/~calkins/math/edrm611/edrm13.htm. Accessed on 08 January 2016
23. R: A Language and Environment for Statistical Computing. https://www.r-project.org/
24. Common Weakness Scoring System. https://cwe.mitre.org/cwss/cwss_v1.0.1.html. Accessed on 08 January 2016
25. Using Exploitability Index. https://technet.microsoft.com/en-us/security/ff943560.aspx. Accessed on 08 January 2016

Defining Objectives for Preventing Cyberstalking

Gurpreet Dhillon[1(✉)], Chandrashekar Challa[2], and Kane Smith[1]

[1] Virginia Commonwealth University, Richmond, USA
{gdhillon,smithkj6}@vcu.edu
[2] Longwood University, Farmville, USA
challacd@longwood.edu

Abstract. Cyberstalking is a significant challenge in the era of Internet and technology. When dealing with cyberstalking, institutions and governments alike have a problem in how to manage it and where to allocate resources. Hence, it is important to understand how individuals feel about the problem of cyberstalking and how it can be managed in the context of cybersecurity. In this paper we systematically interviewed over 100 individuals to interpret their values on cyberstalking. Keeney's [21] value focused thinking approach is then used to convert individual values into objectives which form the basis for planning to curb cyberstalking and for institutions and governments to allocate resources prudently.

Keywords: Cyberstalking · Cyber security planning · Values · Strategic objectives

1 Introduction

Stalking has been well recognized in the academic and practitioner literature; however with the advent of newer technologies such as social media a new threat has emerged, cyberstalking. An increased reliance of individuals on interpersonal contact has resulted in a corresponding increase in possibility of interpersonal intrusion, referred to as cyberstalking [27]. Institutions and government bodies struggle to manage cyberstalking due to a lack of understanding of the phenomenon. The problem question is twofold: First, what are the objectives to ensure protection against cyberstalking. Second, what priority areas should institutions focus on to ensure that cyberstalking is minimized. In this paper we present a comprehensive set of individual value based objectives which can form the basis for strategic planning to prevent cyberstalking. Theoretically we are informed by the value focused thinking concept purported by Keeney [19]. The paper is organized into five sections: introduction, literature pertinent to cyberstalking, the theoretical and methodological aspects of this research, the fundamental and means objectives for minimizing cyberstalking and finally in the limitations and future research directions.

2 A Review of Existing Cyberstalking Literature

The internet is beneficial in connecting us globally on all fronts and is available in nearly every corner of the globe [17]. It is also the cause of many unique crimes such as

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 76–87, 2016.
DOI: 10.1007/978-3-319-33630-5_6

cyberstalking because it is cheap, easy to use, and the anonymity it offers in seeking out victims and avoiding detection [22]. Cyberstalking is a type of crime in which there is no face-to-face contact between victims and offenders [10]. According to McFarlane & Bocij [27], there are four types of cyber stalkers: the Vindictive Cyber stalker, the Composed Cyber stalker, the Intimate Cyber stalker and the Collective Cyber stalker.

In the past, cyberstalking victims have not had success in being recognized as victims by law enforcement agencies due to a lack of enforcement training and expertise [34]. A study that analyzes cyber stalking crimes, legislative intervention measures, and preventative initiatives created specifically to curtail this emerging global crime was undertaken by Pittaro [36]. The study concluded that cyberstalking is a serious and growing problem, but proper training and guidance could allow law enforcement agencies can track the stalkers online [22]. However, educating society is still the most effective approach to bringing awareness about cyberstalking and enacting initiatives prevent this Internet based crime. [34]. California was the first state in the USA to adopt stalking laws in 1989 [51]. Since then stalking laws in general have been adopted elsewhere, but cyberstalking is related to one's behavior in cyberspace as opposed to the physical world [3]. Therefore, it is suggested that there should be an investigation of these regulations and ways to adapt new regulations to apply in cyberspace and how these regulations can help prevent cyberstalking.

While cyberstalking is still in its infancy, it is expected to increase significantly as the Internet becomes more popular [16]. For this reason, there are studies such as the one by Spitzberg and colleagues [8, 9, 41–43] that conducted pilots and introduced the concept of cyber-obsessional pursuit (COP). Further, new research extended these earlier pilot studies to develop and refine measures of cyberstalking victimization [44]. Another study by Goodno [12] examined how differences in state and federal law create gaps in stalking statutes increasing the difficulty in prosecuting all aspects of cyberstalking and suggests ways to close these gaps. Finally, the study also examines the potential issues in criminalizing cyberstalking and how these issues might be resolved by changing the laws so they address the newer cyber security crimes as a result of cyberstalking. Additional work with respect to cyberstalking has sought to develop and adapt a lifestyle–routine activities theory [38] to explain opportunities for victimization in cyberspace environments where traditional conceptions of time and space are less relevant. A related earlier study on the extent and nature of Cyberstalking victimization from a lifestyle/routine activities perspective by Reyns [37] also corroborates the theory. Findings from this study indicate that the number of differing factors such as the number of online social networks an individual owns or low self-control are significant predictors of cyberstalking victimization, suggesting moderate support for lifestyle/routine activities theory in explaining cyberstalking [37].

3 Methodology

In order to identify values one must ask the concerned people [21]. Within the literature, there is a significant variance in the number of individuals that should be interviewed. As an example, Hunter [15] used the interviews of 53 people from two different

organizations to do a content analysis to elicit individual conceptions. However, Phythian & King [35] used two managers who were experts in assessing tender enquiries to identify key factors and rules that influence tender decisions. Additionally, Keeney [21] obtained interviews from over 100 individuals to obtain their values to develop objectives that influenced Internet purchases. For this study, over 100 persons of varying background and experience were interviewed to identify general values for managing cyber stalking related information security.

The following three-step process is used to identify and organize the values that an individual might have with respect to cyberstalking [19]: First, interviews are conducted which elicit the values an individual might have within a decision context. Second, individual values and statements are converted into a common value format, such as an objective oriented statement. Then similar objectives are grouped together in order to form clusters of objectives. Finally, the objectives are then classified as either fundamental to the decision context, resulting in a fundamental objective, or simply a means to achieve the fundamental objectives, or what's called a means objective.

3.1 Identifying Values

To begin, interviews are conducted with the concerned peoples as a process of identifying values. At the beginning of each interview, the purpose is clarified and context and scope of the interview are established. The core objective is to understand the fundamental objectives for preventing cyberstalking. To set the decision context, we emphasize that the scope for eliciting these values is limited only to individuals. After defining the scope of the interview, explanations are provided to the interviewee so that they can understand what 'cyberstalking' is to establish a common understanding. Cyberstalking is thusly defined as 'the use of the internet, email, or other electronic communications devices to stalk another person' [48]. It is made clear to respondents that the goal is to understand values that people might have with respect to cyberstalking. To identify these values four questions are posed about their personal values toward cyberstalking and those of individuals who commit acts of cyberstalking. The questions were: What do you think are your values and wishes in order to prevent cyberstalking? What values might lead you to behave in a certain manner towards cyberstalking? What kind of information do you think people use to engage in cyberstalking? What personal values lead people to use this information for their own benefit while cyberstalking? All questions were open-ended. As individuals can express values differently, so difficulty exists with the quiescent nature of the values, so different probing techniques are used to identify latent values. Keeney [19] suggests words like trade-offs or consequences etc. as useful in making implicit values explicit.

3.2 Structuring Values

Once values have been identified, value structuring and objectives development begins. Step one is that all statements are restated in a common form with duplicates are removed, then common form values are considered and converted into sub-objectives. According to Keeney [21], an objective is constituted of the decision context, an object

and a direction of preferences, which in this case is cyberstalking. With all values systematically reviewed and converted into sub-objectives a number of sub-objectives that deal with a similar issue exists. By carefully reviewing the content of each of these sub-objectives, clusters are developed that group similar ones together and then each cluster of sub-objectives is labeled by its overall theme that becomes the main objective.

3.3 Organizing Objectives

The list of sub-objectives and corresponding clusters initially include both means and fundamental objectives so we must differentiate the two. This is accomplished by repeatedly linking objectives through means–ends relationships then specifying the fundamental objectives. To identify fundamental objectives, the question is asked, 'Why is this objective important in the decision context? [19].' If the objective is an essential reason for interest in the decision context, then the objective is a candidate as a fundamental objective. If the objective is important due its implications with respect to some other objective, then it is a candidate as a means objective. This is termed by Keeney [20] as the 'WITI test.'

4 Objectives for Preventing Cyberstalking

In this section we present fundamental and means objectives for preventing cyberstalking. In our research we found twenty total objectives: five fundamental objectives and fifteen means objectives. In this section we discuss the fundamental and means objectives and how these can collectively contribute to the prevention of cyberstalking.

4.1 Fundamental Objectives for Preventing Cyberstalking

The five fundamental objectives identified in this research include: Protecting Online Interactions; Establishing cyberstalking security procedures, Ensuring technical security, Developing strong value systems and Defining intermediaries to minimize cyberstalking. The fundamental objectives resonate well with what has been defined in the literature and the main characteristic for cyberstalking - repeated event, invasion of personal privacy, evidence of threat and/or fear [44]. Scholars term stalking as a form of *Obsessive Relational Intrusion* (ORI), which is the unwanted pursuit of intimacy [8, 9].

FO1 Protecting Online Interactions. Respondents found protection of online interactions to be defined as both precautionary and regulatory objectives. Exercising caution when meeting people online is fundamental, however it is also important to ensure that protection mechanisms exist in online forums; however the means are addressed in some of our means objectives. A response by one respondent noted: "It is the responsibility of Internet Companies to ensure safety in an online forum through regulation and technical means." In an interesting paper, Chik [6] discusses international cyberstalking regulatory considerations. He notes that there are two basic types of anti-stalking legislations - the list model and the closed model. The list model lists types of offences and

provides certainty, but is rather restrictive. An alternative is the general prohibition model, which is used in some US states and UK. Chik argues that the more open general prohibition model is the favored option [6].

FO2 Establishing Cyberstalking Security Procedures. Respondents felt that good cyberstalking security procedures will go a long way in ensuring security and safety. Cyberstalking security procedures can include an identification of appropriate authentication measures or availability of cyberstalking prevention tools. A respondent noted: "There is no way to tell which site provides adequate security and which one has loose controls, I wish we had a way to do this." Website trustworthiness is an important topic area and has been well researched. At the advent of e-commerce, online vendors were facing similar challenges. Moores and Dhillon [29] found that web assurance seals did help in ensuring a trusting relationship with the consumers. They note: "The relative success of the privacy seals suggests that many sites recognize the issue of privacy and strive to uphold the highest standards. These sites are not the problem. The problem is with those sites that violate their stated obligations, those sites that make no commitment, and those sites that actively seek to exploit the data they collect."

FO3 Ensuring Technical Security. The role of technologies in ensuring security in cyberstalking cannot be underestimated. Unequivocally our respondents made a call for investing in safe browsing technologies and increased abilities to monitor online security settings. Ability to create online filters to block negative behavior was also considered important. One respondent noted: "Now-a-days it is virtually impossible to ensure that the filters are installed properly. People need a high level of competence. Why can't the technologies be made simple and easy to use?" Technical means to ensure online security and its benefits in preventing cyberstalking incidents has been noted by Goldberg [11], who summarizes the problem as one dealing with secure Internet routing. Goldberg notes that secure Internet routing can be achieved through simple cryptographic whitelisting techniques, which can prevent attacks such as prefix hijacks, route leaks, and path-shortening attacks. Some of these attacks are the basis for website compromises, which can then subsequently lead to increased incidents of cyberstalking.

FO4 Developing Strong Value Systems. Early detection of negative behaviors can come about through strong family values and the related social pressures. In a study by Pereira and Matos [33] the complexity of family values is reviewed as well as their impact on cyberstalking. In particular Pereira and Matos found fear following victimization plays a major role in management of cyberstalking [33]. One respondent noted: "I have been cyberstalked. Support from my family was critical in helping me carry on with life."

FO5 Defining Intermediaries to Minimize Cyberstalking. This fundamental objective is somewhat related to the fundamental objective of ensuring secure procedures. Critical to trust forming relationships is the role of intermediaries. Cybersecurity insurance research has suggested that it is indeed possible to minimize threats by appropriately focusing on insurance practices. Pal et al. [32] note: "To alleviate this issue a security vendor can enter the cyber-insurance ecosystem and via a symbiotic relationship

between the insurer can increase its profits and subsequently enable the cyber-insurer to always make strictly positive profits keeping the social welfare state identical. As a special case the security vendor could be the cyber-insurer itself (p. 8)."

4.2 Means Objectives for Preventing Cyberstalking

MO1 Increase Responsibility of Social Media Sites. This objective pertains to organizations responsible for creating, maintaining, regulating and implementing social media sites. These organizations have an obligation to ensure their sites are safe in order to prevent cyberstalking. Many organizations consider this a corporate social responsibility (CSR) and make efforts to shape CSR policies to present themselves as good corporate citizens [28] and the importance of CSR has been emphasized by many in the literature [5].

MO2 Increase Safe Information Sharing. This objective addresses the need for users to have more tools to safely share information on the Internet and social media sites can provide those tools. Types of tools that could be included are increased privacy settings or private web browsing methods. Responses indicate support for this belief such as; "I want more privacy settings and ways to protect my information if I choose to share it."

MO3 Increase Law Enforcement. This objective deals with ensuring useful laws exist to protect online users from cyberstalking. One study analyzed was the police use of Twitter, including the structure of networks and the content of the messages [7]. The study concluded that due to the constraints of police culture, Twitter has been used cautiously as reinforcement for existing means of communication [7]. Responses such as; "law enforcement needs to be more involved in monitoring and policing social media activity" show users want an active law enforcement approach to help prevent cyberstlaking.

MO4 Increase Awareness of Cyberstalking Consequences. This objective addresses the consequences for cyberstalking, specifically making people aware of the negative effects on victims and society as a whole. A survey response supporting this is "as a society we need to increase awareness about the harmful effects of cyberstalking" which speaks to the lack of awareness.

MO5. Minimize Trolling. This objective deals with discouraging people from posting offensive content in their online postings. A study by Hopkinson [14] researched the practice of trolling in online discussion forums and its findings suggest that the definition of trolling varies depending on the discussion topic [14]. The study found a paradoxical view of trolling in that it is considered destructive and have a negative connotation, but cases exist where it can have a positive constructive effect [14].

MO6 Decrease Tracking Ability. This objective deals with ensuring that your current location is unknown to people from whom you wish to remain hidden from. For example, Facebook had a program, which sent messages to users' friends about what they were buying on Web sites; it had to retract this feature after protest from a number of their users due to complaints about sharing without permission [45]. To prevent features like

these from being abused for cyberstalking, the ability of companies and individuals to track people online needs to be minimized.

MO7 Increase Deterrence. This objective deals with the use of punishment as a threat to stop or prevent people from engaging in cyberstalking. Individuals behave rationally to maximize their utility and commit crime when the expected utility of law breaking far exceeds the expected disutility of punishment [18]. So to promote obedience and discourage crime communities should adopt a policy to raise the price of crime. Deterring cyberstalking was a common response, for example, one of our respondents said; "Well-defined laws and stricter enforcement of cyber stalking laws would help prevent cyberstalking."

MO8 Ensure Online Social Responsibility. Organizations and individuals alike have a significant stake in achieving this objective to prevent cyberstalking. There are conflicting views from certain studies whether "Doing Good Always Leads to Doing Better" [39]; however organizations and individuals should proactively take responsibility for making online experiences positive by following basic and fundamental norms of conduct and behavior.

MO9 Personal Accountability. Accountability protects public health and safety, facilitates law enforcement, and enhances national security, but it is more than a bureaucratic concern for corporations, public administrators, and the criminal justice system [2]. In our study we found that respondents have given significant importance to this aspect where one respondent said, "I believe each person is responsible for taking steps towards preventing cyberstalking. That means being mindful of what personal information you share about yourself on the Internet."

MO10 Increase Ability to Control Personal Information. Users desire the ability to control their personal information; how it is shared, stored and distributed over the Internet. Information available online about consumers is striking and the media is filled with horror stories about the misuse of personal information, such as the availability of information most people consider confidential like social security numbers or their home location [40]. Many respondents felt this way with one responding; "I want as many options on social media as possible to prevent as much personal information from being publicly available."

MO11 Ensure Monitoring of Children. This objective deals with the ability to monitor children's online activity and behavior. This is a difficult objective that is highly complex. For example, a national study in Great Britain on children and their parents used focus group interviews and observation of children's use of the internet to reveal the following: Parents seek to manage their children's internet use, but face challenges in helping their children use internet safely. Disagreement between parents and children exists as most children do not want restrictions and have take measures to hide their online activity from their parents demonstrating a gap in the understanding between parents and children on these issues [25]. Their policy recommendations were; direct

children and young people towards valuable content, develop online advice resources with young people etc. [25].

MO12 Reduce Opportunities for Online Victimization. This objective emphasizes the importance of safe browsing and online behavior in order to reduce the opportunity an offensive act can be undertaken by someone. For example, cyberbullying is one major issue in schools and communities due to the emotional, psychological, and even physical harm to which victims can be subjected. One study looked at general strain theory to identify the emotional and behavioral effects of cyberbullying victimization [13]. Data collected indicated that cyberbullying is a potent form of stress that may be related to school behavior problems and delinquent behavior offline [13]. Another study from a national survey of teenagers in the UK (N = 789) analyzed the demographic factors that influence skills in using the Internet and then sought to determine whether these skills make a difference to online opportunities and online risks [26]. Findings show that those who take up more opportunities encounter more risks and vice versa. Further, those groups inclined to gain more opportunities also encounter more risks [26].

MO13 Increase Regulation of Online Social Networks. This objective deals with agencies and government organizations monitoring online social networks and determining the rules and actions that need to be taken to prevent cyberstalking. One study investigated a sample (n = 704) of college students to understand online disclosure and withdrawal of personal information [47]. Findings show little to no relationship between online privacy concerns and information disclosure on online social network sites as students manage unwanted audience concerns by adjusting profile visibility and using nicknames but not by restricting the information within the profile [47]. This behavior suggests that people can still easily gain access to all the free information on Internet, hence why our study suggests social network organizations adopt various countermeasures.

MO14 Increase Mental Health Screening. Mental health can adversely influence one's ability and judgment to conduct themselves properly online. A survey study of 371 British students showed that 18.3 % of the sample was considered to be pathological Internet users, whose excessive use of the Internet was causing academic, social, and interpersonal problems [30]. This would lead one to consider that Internet usage, cyberstalking and mental health are a connected and important area of concern.

MO15 Cyberstalking Education. Cyberstalking and its negative affects are not well known to many people. One study of note was done using students from two universities, which gathered their responses to a cyberstalking scenario as well as their use and experiences with the Internet [1]. Then the study conducted an analysis and comparison of students who reported having been stalked to those who had been cyberstalked [1]. An interesting finding was that male students were statistically more likely than female students to have been cyberstalked [1]. Additionally, for individuals who were cyberstalked, the stalking perpetrator was most likely to be a former intimate partner [1].

5 Further Research, Limitations and Conclusions

Based on the research presented in this paper, there are three broad categories, which exist for future research opportunities. The first opportunity is that the list of objectives identified in this research can be subjected to psychometric analysis using separate large samples. This can help, for example, in developing a model for measuring cyberstalking by organizations on social media sites. A second opportunity exists for intensive research to be undertaken to establish relationships between particular fundamental and means objectives; however, while Keeney [19] contends that fundamental and means objectives are related and an implicit; logical relationships appear to exist between the fundamental and means objectives, but specific relationships need to be researched. The final opportunity is such that further quantitative work should be carried out to assess how the subscales of means and fundamental objectives relate to each other.

The findings of this research lay a suitable foundation for developing multidimensional measures and protections against cyberstalking. For example, Keeney [21] conducted an extensive study, which interviewed over 100 people to assess their values with respect to Internet commerce. And based on this work, Torkzadeh & Dhillon [46] were then able to develop instruments, which measured factors that influence Internet commerce success. Much in the same way, the research presented within this paper has established values and objectives that would be a basis for measures and protections against cyberstalking. Within the IS domain, many examples exist of research that involves in-depth qualitative research aimed at the development of theoretical concepts which includes research on organizational consequences of IT [31], relationship between IS design, development and business strategy [49] and communication richness [24].

In the cybersecurity field, the topic of cyberstalking is constrained by the absence of well-grounded concepts that are developed in a systematic and a methodologically sound manner as the topic itself is still a newer concept. The fundamental and means objectives that are presented in this paper make a contribution towards the development of theory specific to cyberstalking and measures and protections from it, a largely overlooked IS research stream. This research was only the first step to identify means and fundamental objectives as it relates to cyberstalking values. The next step in this research is to conduct a quantitative study as was done earlier by Torkzadeh & Dhillon [46] to come up with an instrument that measures fundamental objectives as it relates to cyberstalking as there is a need to develop theory that is IS specific [4].

As with most qualitative research, this study is subject to some limitations. The process of identifying values from interview data is largely subjective and interpretive and while as researchers we maintain a professional distance, there is always a possibility that some of our own biases may influence the results; however, we were conscious of this during all three phases. The previous basis for this research and the critical reflections of the interviewee's statements was useful in helping us show how these various interpretations emerged in the research [23]. For this reason, it is believed that being aware of the intellectual biases actually helped us to be objective within our analysis of the data. Further, Walsham [50] recognized this to be an issue when carrying out intensive research and in regard to the role of the researcher wrote; "the choice should be consciously made by the researcher dependent on the assessment of… merits

and demerits in each particular case (p. 5)." It is our goal that in strictly following the value-focused thinking method and being conscious that our interpretations should not serve to influence our research, it can provide confidence in the outcome of this study.

In conclusion, the research presented in this paper examines the relatively unexplored area of cyberstalking in the field of information systems. This qualitative investigation, which used value-focused thinking, revealed 75 sub-objectives, grouped into five fundamental and 15 means objectives, which are essential for developing measures and protections against cyberstalking. The objectives developed in this study are grounded socio-organizationally and provide a way forward in developing measures and protections against cyberstalking. Therefore, this is a significant contribution as previous research in this area is underdeveloped and as such falls short of being able to propose tangible measures and protections against cyberstalking.

References

1. Alexy, E.M.: Perceptions of cyberstalking among college students. Brief. Treat. Crisis Interv. **5**(3), 279 (2005)
2. Allen, A.L.: Why privacy isn't everything: Feminist reflections on personal accountability. Rowman & Littlefield (2003)
3. Basu, S., Jones, R.P.: Regulating cyberstalking. J. Inf. Law Technol. **2**, 1–30 (2007)
4. Benbasat, I.: Editorial note. Inf. Syst. Res. **12**, iii–iv (2001)
5. Bauer, T.: The responsibilities of social networking companies: Applying political csr theory to google, facebook and twitter. In: Tench, R., Sun, W., Jones, B. (eds.) Communicating Corporate Social Responsibility: Perspectives and Practice (Critical Studies on Corporate Responsibility, Governance and Sustainability, Volume 6) Emerald Group Publishing Limited, pp. 259–282 (2014)
6. Chik, W.: Harassment through the digital medium-a cross jurisdictional comparative analysis of the law on cyberstalking. J. Int'l Com. L. Tech. **3**, 13 (2008)
7. Crump, J.: What are the police doing on twitter? Social media, the police and the public. Policy Int. **3**(4), 1–27 (2011). Article 7
8. Cupach, W., Spitzberg, B.: Obsessive relational intrusion and stalking. In: Spitzberg, B., Cupach, W. (eds.) The Dark Side of Close Relationships, pp. 233–263. Erlbaum, Hillsdale, NJ (1998)
9. Cupach, W., Spitzberg, B.: Obsessive relational intrusion: incidence, perceived severity, and coping. Violence Vict. **15**(1), 1–16 (2001)
10. Eck, J.E., Clarke, R.V.: Classifying common police problems: A routine activity approach (Crime Prevention Studies, vol. 16, pp. 7–39). Criminal Justice Press, Monsey, NY (2003)
11. Goldberg, S.: Why is it taking so long to secure Internet routing? Commun. ACM **57**(10), 56–63 (2014)
12. Goodno, N.H.: Cyberstalking, a new crime: Evaluating the effectiveness of current state and federal laws. Mo. Law Rev. **72**(1), 125–197 (2007)
13. Hinduja, S., Patchin, J.W.: Offline consequences of online victimization: School violence and delinquency. J. School Violence **6**(3), 89–112 (2007)
14. Hopkinson, C.: Trolling in online discussions: From provocation to community-building. Brno Stud. Engl. **39**(1), 5–25 (2013)
15. Hunter, M.G.: The use of RepGrids to gather data about information systems analysts. Inf. Syst. J. **7**, 67–81 (1997)

16. Hutton, S.: Cyber stalking. Retrieved Feb. 18, 2006, from National White Collar Crime Center (2003)
17. Jaishankar, K., Uma Sankary, V.: Cyber stalking: A global menace in the information superhighway. ERCES Online Q. Rev. **2**(3) (2005)
18. Kahan, D.M.: Social influence, social meaning, and deterrence. V. Law Rev. **83**(2), 349–395 (1997)
19. Keeney, R.L.: Value-Focused Thinking. Harvard University Press, Cambridge, MA, USA (1992)
20. Keeney, R.L.: Creativity in decision making with value-focused thinking. Sloan Manage. Rev. **35**, 33–41 (1994)
21. Keeney, R.L.: The value of Internet commerce to the customer. Manage. Sci. **45**, 533–542 (1999)
22. Reno, J.: 1999 report on cyberstalking: A new challenge for law enforcement and industry. Retrieved Feb. 18, 2006, from US DOJ Web site (1999). http://www.usdoj.gov/criminal/cybercrime/cyberstalking.htm
23. Klein, H.K., Myers, M.D.: A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Q. **23**, 67–94 (1999)
24. Lee, A.S.: Electronic mail as a medium for rich communication: an empirical investigation using hermeneutic interpretation. MIS Q. **18**, 143–157 (1994)
25. Livingstone, S., Bober, M.: UK children go online: Final report of key project findings (2005)
26. Livingstone, S., Helsper, E.: Balancing opportunities and risks in teenagers' use of the Internet: The role of online skills and Internet self-efficacy. New Media Soc. **12**(2), 309–329 (2009)
27. McFarlane, L., Bocij, P.: An exploration of predatory behavior in cyberspace: Towards a typology of cyber stalkers. First Monday, 8, Retrieved Feb 18, 2006 (2005)
28. Melissa, J.R.: Why Social Media Is Vital to Corporate Social Responsibility (2009). http://mashable.com/2009/11/06/social-responsibility/#lL17q023Caqh
29. Moores, T.T., Dhillon, G.: Do privacy seals in e-commerce really work? Commun. ACM **46**(12), 265–271 (2003)
30. Niemz, K., Griffiths, M., Banyard, P.: Prevalence of pathological Internet use among university students and correlations with self-esteem, the General Health Questionnaire (GHQ), and disinhibition. Cyber Psychol. Behav. **8**(6), 562–570 (2005)
31. Orlikowski, W.J., Robey, D.: Information technology and structuring of organizations. Inf. Syst. Res. **2**, 143–169 (1991)
32. Pal, R., Golubchik, L., Psounis, K., Hui, P.: Will cyber-insurance improve network security? A market analysis. In: INFOCOM 2014 Proceedings IEEE, pp. 235–243. IEEE (2014)
33. Pereira, F., Matos, M.: Cyber-stalking victimization: What predicts fear among portuguese adolescents? Eur. J. Crim. Policy Res. **45**, 1–18 (2015)
34. Petrocelli, J.: Cyber stalking. Law Order **53**(12), 56–58 (2005)
35. Phythian, G.J., King, M.: Developing an Expert System for tender enquiry evaluation: a case study. Eur. J. Oper. Res. **56**, 15–29 (1992)
36. Pittaro, M.: Cyber stalking: An analysis of online harassment and intimidation. Int. J. Cyber Criminol. (IJCC) **1**(2), 180–197 (2007). ISSN: 0974 – 2891
37. Reyns, B.W.: Being Pursued Online: Extent and Nature of Cyberstalking Victimization from a Lifestyle/Routine Activities Perspective; A Dissertation Submitted to the: Graduate School of the University of Cincinnati (2010)
38. Reyns, B.W., Henson, B., Fisher, B.S.: Applying cyberlifestyle-routine activities theory to cyberstalking victimization. Crim. Justice Behav. **38**(11), 1149–1169 (2011)

39. Sen, S., Bhattacharya, C.B.: Does doing good always lead to doing better? Consumer reactions to corporate social responsibility. J. Mark. Res. **38**(2), 225–243 (2001)
40. Sovern, J.: Opting in, opting out, or no options at all: The fight for control of personal information. Wash. L. Rev. **74**, 1033 (1999)
41. Spitzberg, B., Nicastro, A., Cousins, A.: Exploring the interactional phenomenon of stalking and obsessive relational intrusion. Commun. Reports **11**(1), 33–48 (1998)
42. Spitzberg, B., Rhea, J.: Obsessive relational intrusion and sexual coercion victimization. J. Interpersonal Violence **14**(1), 3–20 (1999)
43. Spitzberg, B., Marshall, L., Cupach, W.: Obsessive relational intrusion, coping, and sexual coercion victimization. Commun. Reports **14**(1), 19–30 (2001)
44. Spitzberg, B.H., Hoobler, G.: Cyberstalking and the technologies of interpersonal terrorism. New Media Soc. **4**(1), 67–92 (2002)
45. Story, L., Stone, B.: Facebook Retreats on Online Tracking (2007). www.nytimes.com
46. Torkzadeh, G., Dhillon, G.: Measuring factors that influence the success of internet commerce. Inf. Syst. Res. **13**, 187–204 (2002)
47. Tufekci, Z.: Can you see me now? Audience and disclosure regulation in online social network sites. Bull. Sci. Technol. Soc. **28**, 20–36 (2008)
48. US Attorney General (1999) 'Cyberstalking: A New Challenge for Law Enforcement and Industry. Report from the Attorney General to the Vice President
49. Walsham, G., Waema, T.: Information systems strategy and implementation: A case study of a building society. ACM Trans. Inf. Syst. **12**, 150–173 (1994)
50. Walsham, G.: Interpretive case studies in IS research: nature and method. Eur. J. Inf. Syst. **4**, 74–81 (1995)
51. Zona, M.A., Sharma, K.K., Lane, M.D.: A comparative study of erotomanic and obsessional subjects in a forensic sample. J. Forensic Sci. **38**, 894–903 (1993)

Cyber Infrastructure

Using Process Invariants to Detect Cyber Attacks on a Water Treatment System

Sridhar Adepu(✉) and Aditya Mathur

Singapore University of Technology and Design, Singapore 487372, Singapore
{sridhar_adepu,aditya_mathur}@sutd.edu.sg

Abstract. An experimental investigation was undertaken to assess the effectiveness of process invariants in detecting cyber-attacks on an Industrial Control System (ICS). An invariant was derived from one selected sub-process and coded into the corresponding controller. Experiments were performed each with an attack selected from a set of three stealthy attack types and launched in different states of the system to cause tank overflow and degrade system productivity. The impact of power failure, possibly due to an attack on the power source, was also studied. The effectiveness of the detection method was investigated against several design parameters. Despite the apparent simplicity of the experiment, results point to challenges in implementing invariant-based attack detection in an operational Industrial Control System.

Keywords: Attack detection · Cyber attacks · Cyber physical systems · Industrial control systems · Secure water treatment testbed

1 Introduction

An experimental investigation, referred to as EXP, was undertaken with the long term goal of designing robust defense mechanisms for an Industrial Control System (ICS) and to improve its resiliency. A short term goal in EXP is to understand how to detect cyber attacks against an ICS using state invariants across data obtained by a Programmable Logic Controller (PLC) from two or more sensors. Such an understanding leads to the inclusion of effective detection mechanisms inside process controllers and the addition of control actions when an attack is detected thereby improving system resiliency. The experiments were performed to understand the effectiveness of the proposed detection method in an operational mini-water treatment testbed, referred to as SWaT (Secure Water Treatment), that produces 5 gallons/minute of treated water.

Invariant: An "invariant" is a condition among "physical" and/or "chemical" properties of the process that must hold whenever an ICS is in a given state. Together, at a given time instant, measurements of a suitable set of such properties constitute the observable state of SWaT. In SWaT, these properties are measured using sensors and captured by the PLCs at programmable time instants,

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 91–104, 2016.
DOI: 10.1007/978-3-319-33630-5_7

(set to 0.1-s in EXP). A few key advantages of invariant-based attack detection, implemented in a PLC, are as follows.

1. *Implementation context*: Detection method is implemented as a procedure and integrated directly into the PLC.
2. *Physical constraints*: The invariants are local to a PLC and based on the physics or the chemistry of the sub-process being controlled by the PLC.
3. *Detection method*: Detection is designed without reference to attacks and hence is *attack agnostic*. It is based on state related conditions that must hold either always during system operation, or when one or more components of the system is in a given state. For example, the level of water in a tank must be always between its lowest and highest points; this invariant must always hold during normal system operation. However, a pump must be in the ON state only when the source tank has water and the destination tank is not full; this is a state dependent invariant.
4. *Network traffic*: There is no additional load on the ICS communications network.

Research focus: RF1: Methods for detecting single-point cyber attacks based on invariants derived from physical properties of the process. RF2: effectivenessEffectiveness of the invariant-based detection methods. RF3: complexity-Complexity and scalability of the detection methods and their impact on the operation of a PLC.

Related work: In a survey [8] the detection techniques are classified as follows: misuse/signature-based intrusion detection, anomaly-based intrusion detection, and statefull protocol analysis. The process invariants based approach proposed in this paper does not fall in any of these three categories. Further, as implemented in the case study described here, the proposed approach differs from those mentioned above in several ways including the fact that it does not use network-based anomaly detection. Pros and cons of the proposed approach against others are discussed in Sect. 4. A common aspect of the techniques described in [8] is that they are employed in the network used for communications among the PLCs and SCADA. The primary goal of these techniques is to detect any intrusion into the ICS by analyzing network traffic from different points of view. For example, in [3] security specifications are defined for smart meters and a security policy for the Advanced Metering Infrastructure.

The use of invariants in CPS is not new. In [12] the authors used invariants as a unified knowledge model for CPS. Some authors refer to invariants as "attack symptoms" and have used these to detect attacks in intrusion detection [9]. Stability of smart-grid has been studied using invariants [5]. The key contribution in this paper is the derivation and use of invariants using the physics of a water treatment plant and its application in detecting novel attacks.

A specification-agnostic technique for the detection of cyber attacks in PLCs is described in [7]. It is noted that this technique does not fall in any of the three categories mentioned above. Data is collected from the PLCs via network monitoring, and an autoregressive method is used to model specific process variables.

Such a model is then used to detect whether the value of a process variable is suspicious. Another technique uses the physics of an ICS, using conditions similar to invariants, to detect cyber attacks [10]. Here the authors analyze network traffic focusing on "harmful" command streams. Physical constraints are integrated into an intrusion detection framework. An example of a boiler is described where out of bound values are checked against predefined constraints.

Significant work exists in detecting anomalies in network traffic in ICS across PLCs, sensors, actuators and SCADA subsystems. One such technique is based on CUSUM used for change point detection. This non-parametric method has been applied to detect network intrusions [14]. While these techniques are found effective in environments in which they were assessed, in EXP it was decided to instead use only the process property based invariants to detect anomalies arising due to a cyber attack. Doing so avoids making assumptions on probability distributions of process data. Indeed, making use of invariants is perhaps appropriate when a real or simulated process is available for experimentation, and not necessarily when only data from such process is available as for example in [7,15].

Intermittent cyber attacks: Studies on the impact and detection of intermittent, or pulse, attacks on networks have been reported. In [16] the authors considered the impact of low-rate denial of service (LDoS) attacks on networks. A wavelet based method was proposed for detecting such attacks. In [13] the authors focused on pulsing DoS attacks and their impact on peak bandwidth. The experiments described in this paper are aimed at investigating how intermittent cyber attacks on an ICS, and not on networks, can lead to undesirable behavior and the difficulty of detecting them using invariants. The authors of the current study are not aware of any experiment that investigates the impact of intermittent attacks on ICS.

Contributions: (a) Invariant-based approach for attack detection in a specific ICS. (b) Dependence of the invariant-based approach on the system state, and several other parameters, when an attack is launched. (c) Impact of attack detection study on the design of the software and hardware of a specific ICS.

Organization: Section 2 describes the method used and the experiments conducted. A brief introduction to SWaT is in this section. Results from the experiment appear in Sect. 3. Discussion on various aspects of attack detection in an ICS and design challenges, are in Sect. 4. Conclusions and plans for further experimentation are in Sect. 5.

2 Method

2.1 Context: The SWaT testbed

SWaT is a fully operational scaled down water treatment plant for research in the design of ICS resilient to cyber and physical attacks. In a small footprint producing 5 gallons/minute of doubly filtered water, this testbed mimics large

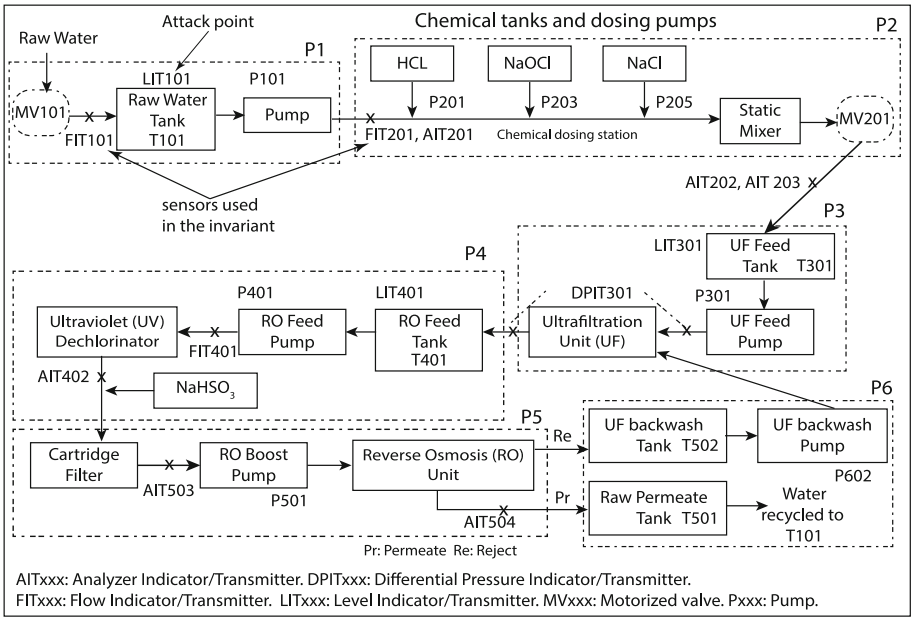


Fig. 1. Water treatment in SWaT: P1 through P6 indicate the six stages in the treatment process. Arrows denote the flow of water and of chemicals at the dosing station.

modern plants for water treatment such as those found in cities. The testbed is available for investigating the response to cyber-attacks and for conducting experiments with novel designs of physics-based and other attack detection and defense mechanisms.

Water treatment process: The treatment process (Fig. 1) in SWaT consists of six distinct and cooperating sub-processes P1 through P6. Each sub-process, referred to as a *stage*, is controlled by an independent PLC. Thus, six PLCs work in concert to control the entire process. Control actions are based on the system state estimated by the PLCs using data from sensors.

Stage P1 controls the inflow of water to be treated, by opening or closing a motorized valve that connects the inlet pipe to the raw water tank. Water from the raw water tank is pumped via a chemical dosing station (stage P2) to another UF (Ultra Filtration) feed water tank in stage P3. A UF feed pump in P3 sends water via the UF unit to RO (Reverse Osmosis) feed water tank in stage P4. Here an RO feed pump sends water through an ultraviolet dechlorination unit controlled by a PLC in stage P4. In stage P5, the dechlorinated water is passed through a 2-stage RO filtration unit. The filtered water from the RO unit is stored in the permeate tank and the reject in the UF backwash tank. Stage P6 controls the cleaning of the membranes in the UF unit by turning on or off the UF backwash pump.

Communications: Each PLC obtains data from sensors associated with the corresponding stage, and controls pumps and valves in its domain. Ultrasonic level sensors in each tank inform the PLCs of water level in the corresponding tank. Several other sensors are available to check the physical and chemical properties of water flowing through the six stages. PLCs communicate with each other through a separate network. Communications among sensors, actuators, and PLCs can be via either wired or wireless links; manual switches allow switch between the wired and wireless modes.

Attacking SWaT: The wireless network in SWaT connects PLCs to sensors, actuators, and to the SCADA server and an engineering workstation. Attacks that exploit vulnerabilities in the protocol used, and in the PLC firmware, are feasible and could compromise the communications links between sensors and PLCs, PLCs and actuators, among the PLCs, and between PLC and SCDA and the historian. Having compromised one or more links, an attacker could use one of several strategies to send fake state data to one or more PLCs, or simply do reconnaissance for a possibly subsequent attack.

2.2 Experiments

System State. *Components and states*: Each PLC in SWaT controls one or more actuators such as a pump. A *Local Component Set* consists of components whose state is directly sensed by a PLC and the actuators that the PLC controls. The actions of a PLC depend on the state of the components in its LCS, and might also depend on the state of components in the LCS of one or more other PLCs. In the latter case a PLC can communicate with the other PLCs via the communication network to retrieve the required state data. The union of LCS for each PLC constitutes the *Global Component Set (GCS)*. The *local* state of SWaT is comprised of the respective observable states of components under direct control of a single PLC, i.e., that of its LCS. A collection of local observable states of all six PLCs in SWaT constitutes its *global* state. It is important to note that only the observable properties of the process and its components are included, and used for attack detection, in the local and global states.

State set: As shown in Table 1, three distinct local observable states were selected for the experiments reported here. These are the states when an attack is launched. In $\mathbf{S}_0^1$ the level of Tank T101 is constant during the attack. In $\mathbf{S}_0^2$ the level of T101 is increasing which happens when the level goes below a pre-determined value. In $\mathbf{S}_0^3$ the level in Tank T101 is decreasing which could happen because tank T201 in the sub-process in P3, is below a pre-determined level. These three states of T101 are marked as A, B, and C in Fig. 2.

Attack Design. *Attack types*: In this work the focus is on attack detection assuming that an attacker succeeds in launching it. While paths through a system used by an attacker to enter by exploiting a system vulnerability, e.g., design flaw in a PLC [6], is an important topic, it is not the focus of this work. The number of possible attacks on an ICS is exorbitantly large. Three distinct types

Table 1. State of P1 at the time of attack launch.

System State	Sensor[a]	State	Description
$\mathbf{S}_0^1$	**LIT101**←	HH[b]	T T101 is full
	MV101	Closed	No flow into T101
	P101	OFF	No outflow from T101
$\mathbf{S}_0^2$	**LIT101**←	L	T101 is not full
	MV101	open	Water flows into T101
	P101	OFF	No outflow from T101;
$\mathbf{S}_0^3$	**LIT101**←	H	T101 is full
	MV101	Closed	No flow into T101
	P101	ON	Flow out of T101

[a]Sensors not listed are not used during attack detection. As marked, LIT101 is attacked.
[b]Tank states: HH=1000 mm, H=800 mm, L=500 mm, LL=250 mm.

of attacks were selected, namely, fixed bias (FB), fixed bias intermittent (FBI), and variable bias intermittent (VBI). The choice of these three attack types was motivated by (a) attacks used in [4] to study the effectiveness of a statistical technique in detecting cyber attacks in a chemical process, and (b) a series of thought experiments aimed at designing attacks that might be difficult to detect in certain specific states of SWaT.

Stealthy attacks: A stealthy attack on a CPS is one that remains undetected unless special detection mechanisms are in place. Such attacks have been studied in the context of CPS [4]. While non-stealthy attacks are possible in SWaT, all

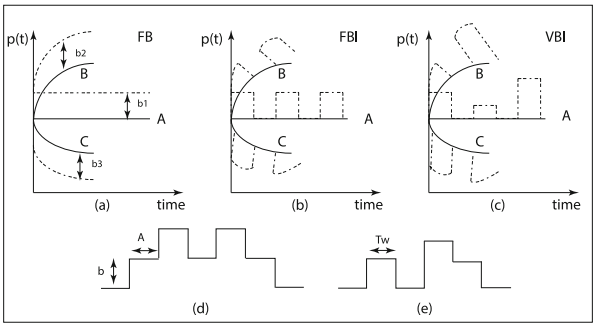


Fig. 2. Attack types (a) Fixed Bias (FB), (b) Fixed Bias Intermittent (FBI), and(c) Variable Bias Intermittent (VBI), superimposed on physical property p(t) as indicated by the dotted lines. In FB and FBI, the attacks follow the change in property p(t) in an attempt to avoid detection. (d) FBI; bias and pulse width are fixed. (e) VBI; bias b and pulse width T_w are varied.

Table 2. Summary of states and parameters used in the experiments.

State or parameter	Values	Comments
Actuator and tank states	$\{\mathbf{S}_0^1, \mathbf{S}_0^2, \mathbf{S}_0^3\}$	States of various components of SWaT set prior to launching an attack; details are in Table 1
Power outage	{NPF, PF}	Attack launched during normal operation (NPF) and immediately before or after power outage (PF)
Attack type	{FB, FBI, VBI}	Fixed bias (FB), fixed bias intermittent (FBI), and variable bias intermitent (VBI)
Bias (b)	{{>4mm}, {<4mm}}	Bias used each of the three attack types
Pulse width (T_w)	{8, 13}	Width of the attack pulse (in seconds)(Fig. 2 used in FBI and VBI
Detection duration (n)	10	Number of sensor readings used prior to announcing a decision on whether the sensor is under attack or not

attacks launched in the experiment reported here were stealthy as the system software and hardware was unable to detect these until a system damage had occurred [1].

Attack: As shown in Table 2, several attacks types were examined within each state. For example, in NPF (No Power failure), an attack is launched after the PLC has initialized itself with the current state of its local components. In PF, an attack is launched just prior to the PLC starting to initialize itself and hence does not *yet* have information about its local components. In addition, the bias b used in an attack as well as the attack pulse width (T_w) were varied. The duration of attack detection (n), i.e., the number of sensor readings used to compute the invariant, was fixed at 10, where one reading is obtained every second. A selection of one value from each set in Table 2 served to define an attack. For example, an FBI attack was launched when SWaT was in state $\mathbf{S}_0^1$, attack bias was larger than 4, pulse width was 8, and the system was operating in a stable state, i.e., there was no power outage in the immediate past. Thus, a total of $3 \times 2 \times 3 \times 2 \times 2 \times 1 = 72$ experiment combinations exist. Additionally, a large number of trial runs had to be conducted to obtain reasonable values of bias, width of the attack pulse, and parameter ϵ mentioned later in Eq. 4.

Attack procedure: The following general procedure was used to launch cyber attacks in EXP.

1. Identify the *tag* to be manipulated in the attack; a tag is a memory location where a PLC saves the received sensor data.
2. Compromise the wireless link between the SCADA computer and PLCs.
3. Manipulate the *tag* by setting its value different from that received by the PLC. In the absence of any hardware or attack detection logic in the PLC code, the PLC assumes the manipulated value to represent the true state of the component that corresponds to the sensor whose output is manipulated.

2.3 State Estimation

Let x denote property p and y its measurement. $y(k)$ denotes the sensor measurement for $x(k)$ at instant k. $\hat{x}(k)$ is an estimate of $x(k)$. In the absence of sensor errors and no cyber attacks, $\hat{x}(k) = x(k) = y(k)$. In EXP the water level in tank T101 was considered as p. The level in T101 is measured by sensor LIT101 (Fig. 1) that was assumed to be under attack. Sensors FIT101 and FIT201 measure, respectively, water flow into and out of T101. These flow rates are denoted as $u_i(k)$ for inflow, and $u_o(k)$ for outflow. In SWaT, each PLC obtains sensor data at 0.1 second intervals though for detecting an attack data was sampled from LIT101 every second as smaller sampling intervals did not offer any benefit in attack detection in trial runs.

In EXP the attacker intent was to cause tank T101 to overflow and degrade the performance of SWaT so that it produces less water than its normal capacity. This intent was to be realized by attacking the level sensor LIT101 that measures and reports $x(k)$ to PLC 1 in stage P1. $\hat{x}(k+1)$ was computed using methods M1 and M2 described next.

Method M1: Open loop: In this method $\hat{x}(k+1)$ is estimated using $\hat{x}(k)$ as follows.

$$\begin{aligned} \hat{x}(0) &= y(0) \\ \hat{x}(k+1) &= \hat{x}(k) + \alpha(u_{in}(k) - u_{out}(k)) \end{aligned} \tag{1}$$

where α converts flow rate to the change in the level of T101 using the physical dimensions of the tank.

Method M2: Closed loop: In this method $\hat{x}(k+1)$ is estimated using $y(k)$ as follows.

$$\begin{aligned} \hat{x}(0) &= y(0) \\ \hat{x}(k+1) &= y(k) + \alpha(u_{in}(k) - u_{out}(k)) \end{aligned} \tag{2}$$

Note that in M1 the estimate of tank level is updated using previous state estimate with the initial value obtained from the sensor L101. In M2 the tank level is estimated using sensor values. In trial experiments it was observed that the open loop method is not suitable for deriving an invariant as the method does not account for the change in system dynamics. Hence M1 was abandoned.

2.4 Invariants

At time instant $k+1$, the water level in T101 depends on the level at time k and the inflow and outflow at instant k. This relationship is captured in the following idealized model of the tank,

$$x(k+1) - x(k) = \alpha(u_i(k) - u_o(k)), \tag{3}$$

where (3) assumes perfect sensors which is not true in practice. Hence, to derive a practically usable invariant, SWaT was run several times without any attacks to estimate the mean μ_d and the standard deviation σ_d of $d = (\hat{x}(k) - y(k))$ over several runs, i.e., the mean and variance of the difference between the estimated tank level $\hat{x}(k)$ and its measured value $(y(k))$. In these runs $\hat{x}(k)$ was computed using the closed loop method M2 as in Eq. 2. Based on Eq. 3, the statistics obtained experimentally, and converting the true states to their estimates, the following conditions were derived to test whether or not sensor LIT101 is under attack.

$$\frac{\sum_{i=1}^{n}(\hat{x}(i) - y(i))}{n} > \epsilon, \quad \text{under attack,} \tag{4}$$

$$\leq \epsilon, \quad \text{normal.} \tag{5}$$

In the conditions above, the average of the difference between the estimated and the measured tank levels is tested against ϵ. Thus, a decision whether or not LIT101 is under attack is taken from n sensor readings. Selection of n ought to be done carefully as it impacts the detection effectiveness. In EXP n, was set to 10. As described earlier, based on trial runs of SWaT without attacks, ϵ was set to 0.55. Code that implements attack detection using the invariant in Eqs. 4 and 5 was added to the control algorithm already built into PLC 1.

3 Results

Data obtained from the experiments, and its analyses, are reported in the following.

3.1 Detection Effectiveness and Impact

In each of the three states when the attacks were launched (Table 1), there were six attack modes, namely, FB-PF, FB-NPF, FBI-PF, FBI-NPF, VBI-PF, and VBI-NPF. The attack detection results are summarized as follows.

1. Detection of attacks launched in FB-PF mode, and the system response, was found to be independent of T_w. However, it does depend on bias b and the initial state when the attack was launched. This happens because the PLC looses prior state data $(y(k))$ from LIT101. Hence, it initializes $\hat{x}(0)$ by obtaining $y(0)$ from LIT101 which is under attack. Thus, the initial state estimate is the tank level indicated by the attacker, i.e., (actual tank level+b), and not

the actual tank level for T101. This happens regardless of the initial states of SWaT, i.e., $\mathbf{S}_0^1$, $\mathbf{S}_0^2$, and $\mathbf{S}_0^3$. From this point onwards, the PLC computes the remaining values of $\hat{x}(k+1)$ using the incorrect $\hat{x}(0)$. As explained next, the response of SWaT now depends on the initial state and $\hat{x}(0)$. As there exist several variations of the attack depending on the value of b, four sample cases are discussed next.

Case 1: Initial state=$\mathbf{S}_0^1$, $\hat{x}(0)$<L. [*Attack not detected; tank overflow*] PLC opens MV101 and water starts flowing into T101. The invariant in Eq. 4 is computed over the following 10 seconds. However, the condition for attack detection is false and hence the attack is not detected. After some time T101 overflows.

Case 2: Initial state=$\mathbf{S}_0^1$, $\hat{x}(0)$>L. [*Attack not detected; performance degradation*] In this case the PLC does not open MV101. Assuming that the ultrafiltration process is active, the level in T301 is reducing. When this level falls below H then pump P101 will be started by PLC 1. This causes tank level of T101 to decrease gradually while LIT101 continues to report the injected value. Eventually T101 becomes empty, P101 is stopped and water stops flowing into T301. This will cause the level in T301 to drop and eventually stop P301 and the ultrafiltration process. This event will eventually lead to the RO process stopping as there is no water to be filtered. Thus, this attack leads to a reduction in system performance.

Case 3: Initial state=$\mathbf{S}_0^2$, $\hat{x}(0)$<L. [*Attack not detected; overflow*] Given that the bias is fixed and the attacker is following the water level trajectory in the tank, MV101 will continue to be open until LIT101 indicates HH. However, the attacker can control b such that the level indicated by attacked LIT101 is much lower than the tank level at attack launch. Thus, if tank level is $L_1 < L$ immediately prior to the attack, and $b = -200$, then T101 will overflow because the buffer in T101, beyond level HH, is only 100 mm. In this attack scenario, if $b > 0$ then the attack will not be detected and there will be no overflow as the PLC will shut MV101 when the injected value of LIT101 reaches HH.

Case 4: Initial state=$\mathbf{S}_0^3$, $\hat{x}(0)$>H. [*Attack not detected; no damage*] In this case when the injected LIT101 value reaches L, MV101 will open and the tank level will start to rise. Thus, the attack is not detected while the level is decreasing. When the level begins to rise, the attacker will need to change the bias to remain undetected as the scenario now is similar to the one in case 3. For other values of b the attack is not detected and no damage done.

2. In the remaining five modes with b>4, all 30 attacks were detected. Several experiments were performed to investigate the impact of b and T_w. With b=3 and T_w=8, 40 % of the attacks launched were detected and the remaining not detected. With b= 3 and T_w= 13, 10 % of the attacks were detected, and the remaining not detected. With b<3 none of the attacks were detected.

3.2 Selection of n and ϵ

The experiments revealed the importance of selecting appropriate values of n and ϵ used in the invariant to decide whether or not LIT101 is under attack. Data from the experiments was used to investigate the relationship between n, ϵ, and false alarms. Such investigation is also needed, and is underway, to understand the relationship between n, ϵ and the attack detection effectiveness.

Figure 3 shows how false alarms depend on n and ϵ. The data in the plots in Fig. 3 was collected over a period of 40 minutes of SWaT operation in NPF mode. It is clear that the false alarms decrease as n and ϵ increase. For $\epsilon = 0.5$, the best value of n lies around 6. Similarly, for $n = 3$ the best value of ϵ is around 0.65. Thus, $n = 6$ and $\epsilon = 0.65$ appears to be the best combination if minimization of false alarm rate is the objective. However, attack detection rate also reduces with increasing n as well as with increasing ϵ. Thus, additional experiments need to be conducted to find optimal values of n and ϵ that maximize the attack detection rate while minimizing the false alarm rate.

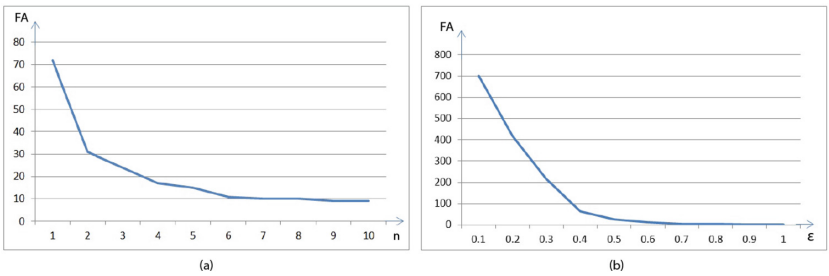


Fig. 3. False alarms (a) vs n for $\epsilon = 0.5$, (b) vs ϵ for $n = 3$.

4 Discussion

RF1: Two methods, namely, M1 and M2 are proposed for computing invariants to detect cyber attacks in ICS. Both methods can be used to derive invariants from the properties of the process that could be attacked. Experiments were conducted using only M2 as M1 was not found suitable for use in a system with changing dynamics.

RF2: In all experiments, attacks were launched when SWaT was in one of three system states, namely, $\mathbf{S}_0^1$, $\mathbf{S}_0^2$, and $\mathbf{S}_0^3$. Within each set there were two sub-states: without system reset (NPF) and soon after system reset (PF) such as what might happen after power is removed from the system. In NPF, the attack was launched soon after SWaT was started but all PLC's had executed their respective control algorithms at least once. In PF, the attack was launched and the PLC reset prior to it completing n cycles of code execution, where n is the attack detection

window. Results indicate that attack detection becomes challenging when the attack is launched during PLC reset. Resetting a PLC causes it to loose the prior state information. Thus, if an attack detection algorithm must initialize its knowledge of the system state soon after the reset operation then doing such initialization from the sensor ought to be avoided as the attacker might have already compromised the sensor. Another approach could be to obtain the initial state of the process from the historian where all sensor data is saved. Certainly, this approach is advisable assuming that the sensor value saved in the historian is the true state value and not the one sent by an attacker. Thus, even in the case of a perfect attack detection algorithm, sensor data sent to the historian could be from an attacker if the attack was not detected prior to the reset operation.
RF3: Derivation and implementation of the invariants in Eqs. 4 and 5 was relatively straightforward. The invariant was implemented in Structured Text, a commonly used programming language for PLCs. Adding this code to the existing code in PLC 1 had negligible impact (at most 3milli-seconds) on the time to execute one scan cycle. It is not clear what will be the impact of adding code that implements traditional schemes, e.g. Kalman filter or Lurenberger observer, for removing noise from sensor data before it is used in computing the invariant. However, one could complement a PLC with additional hardware, say based on FPGA, that encodes the invariant and is connected to the main SCADA workstation to send alerts. Doing so would not add computational load to the PLC.

Generality of invariant based detection: Invariants derived in EXP capture the dynamics of tank level. In SWaT, several other invariants exist, but were not derived. These include, invariants that relate water properties such as pH, ORP, and conductivity as water flows across the chemical dosing station, UF, and RO units. Such invariants are important to detect cyber attacks aimed at affecting properties of water within a plant as well that coming out of a plant. However, such invariants also depend on the chemical and physical properties of the units involved. For example, the relation between the pH of water entering the RO unit and that coming out as permeate, depends on the physical properties of the membranes in RO. The time dependent nature of such properties requires tuning of any parameters used in the invariant to reduce false positives. It is evident that while for a specific sub-process in any ICS, one can define a process invariant, many such invariants exist in one plant.

Multiple point attacks: The cyber attacks considered in EXP are single point. The detection mechanism proposed in this work can be thwarted when the attacker has access to the sensors used in the invariants. For example, in Eqs. 1 and 2, sensors FIT101 and FIT201 are used to measure the inflow and outflow rates. The detection method can fail when these two flow sensors are compromised. Thus, to detect attacks on multiple sensors, one needs additional state information that will likely be derived from one or more sub-processes in an ICS that are not under attack. Doing so may or may not detect attacks, and even when detected, the detection might be delayed based on the state of the sub-processes.

Emerging design challenges: The following key design parameters were identified in EXP: n and ϵ in Eq. 4; A_w: number of contiguous sensor inputs used by the control logic in the PLC to decide whether to initiate an action on an actuator; T_w: pulse width; and b: bias. n, A_w, and ϵ can be controlled in the software that implements the attack detection and control algorithms in a PLC. However, T_w and b are controlled by the attacker. Given the knowledge of n, ϵ, and A_w, the attacker can adjust T_w and b and succeed in causing SWaT malfunction, and especially so if the attack is launched before a PLC has been able to reset itself, i.e., in the PF mode. Parameter R_w can be controlled to some extent by selecting appropriate actuators. Thus, a key research question arises: How should one determine the most appropriate values of A_w, n, and ϵ given the uncertainty in T_w and b? This question assumes important in light of the fact that sensors could lead to spurious data and that the error profile of a sensor may change over time requiring the D_w and A_w to be retuned.

5 Conclusions and Future Work

An experimental investigation was undertaken to understand the effectiveness of attack detection using process invariants. The experiments were performed on an operational water treatment system. One water level sensor was selected as the target of the attacker. An invariant was derived from the dynamics of water flow into and out of a tank. Results from the experiments clearly indicate the strengths and limitations of the invariant-based approach for attack detection. Several ICS design parameters were identified. The appropriate values of these parameters can be selected by the designer and depend on system dynamics. Selection of parameter values is a subject of study by itself and needs to be taken seriously to avoid false positives. Note that it is possible for an attacker to bypass the detection method if the parameter values are known.

While the experiments indicate that the invariant-based detected method is effective in detecting a variety of attacks, no claims are made regarding the detection effectiveness in other domains such as power and transportation. A theoretical study is needed to better understand why physics-based invariants are, or are not, able to detect the attacks.

Acknowledgements. Kaung Myat Aung for assistance in conducting the experiments. This work was supported by research grant 9013102373 from the Ministry of Defense and NRF2014-NCR-NCR001-040 from the National Research Foundation, Singapore.

References

1. Adepu, S., Mathur, A.: An investigation into the response of a water treatment system to cyber attacks. In: Proceedings of the 17th IEEE High Assurance Systems Engineering Symposium, Orlando, January 2016

2. Beaver, J., Borges-Hink, R., Buckner, M.: An evaluation of machine learning methods to detect malicious SCADA communications. In: 12th International Conference on Machine Learning and Applications (ICMLA), vol. 2, pp. 54–59, December 2013
3. Berthier, R. Sanders.: Specification-based intrusion detection for advanced metering infrastructures. In: 17th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 184–193, October 2011
4. Cárdenas, A.A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., Sastry, S.: Attacks against process control systems: Risk assessment, detection, and response. In: ACM Symposium on Information, Computer and Communications Security (2011)
5. Choudhari, A., Ramaprasad, H., Paul, T., Kimball, J., Zawodniok, M., McMillin, B., Chellappan, S.: Stability of a cyber-physical smart grid system using cooperating invariants. In: 2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC), pp. 760–769, July 2013
6. ICS-CERT Advisories. https://ics-cert.us-cert.gov/advisories
7. Hadžiosmanović, D., Sommer, R., Zambon, E., Hartel, P.H.: Through the eye of the PLC: Semantic security monitoring for industrial processes. In: Proceedings of the 30th Annual Computer Security Applications Conference, pp. 126–135, New York, NY, USA, ACM (2014)
8. Han, S., Xie, M., Chen, H.-H., Ling, Y.: Intrusion detection in cyber-physical systems: Techniques and challenges. IEEE Syst. J. **8**(4), 1049–1059 (2014)
9. Hsiao, S.-W., Sun, Y., Chen, M.C., Zhang, H.: Cross-level behavioral analysis for robust early intrusion detection. In: IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 95–100, May 2010
10. McParland, C., Peisert, S., Scaglione, A.: Monitoring security of networked control systems: It's the physics. IEEE Secur. Priv. **12**(6), 32–39 (2014)
11. Niazi, R.H., Shamsi, J.A., Waseem, T., Khan, M.M.: Signature-based detection of privilege-escalation attacks on Android. In: 2015 Conference on Information Assurance and Cyber Security (CIACS), pp. 44–49, December 2015
12. Paul, T., Kimball, J., Zawodniok, M., Roth, T., McMillin, B.: Invariants as a unified knowledge model for cyber-physical systems. In: IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–8, December 2011
13. Rasti, R., Murthy, M., Weaver, N., Paxson, V.: Temporal lensing and its application in pulsing denial-of-service attacks. In: IEEE Symposium on Security and Privacy (SP), pp. 187–198, May 2015
14. Tartakovsky, A., Rozovskii, B., Blazek, R., Kim, H.: A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. IEEE Trans. Signal Process. **54**(9), 3372–3382 (2006)
15. Thatte, G., Mitra, U., Heidemann, J.: Parametric methods for anomaly detection in aggregate traffic. IEEE/ACM Trans. Netw. **19**(2), 512–525 (2011)
16. Wu, Z.-J., Zhang, L., Yue, M.: Low-rate DoS attacks detection based on network multifractal. IEEE Trans. Dependable Secure Comput. **PP**(99), 1–10 (2015)

Expression and Enforcement of Security Policy for Virtual Resource Allocation in IaaS Cloud

Yanhuang Li[1,2(✉)], Nora Cuppens-Boulahia[2], Jean-Michel Crom[1], Frédéric Cuppens[2], and Vincent Frey[1]

[1] Orange Labs, 4 rue du Clos Courtel, 35510 Cesson-sévigné, France
{yanhuang.li,jeanmichel.crom,vincent.frey}@orange.com
[2] Télécom Bretagne, 2 rue de la Châtaigneraie, 35510 Cesson-sévigné, France
{nora.cuppens,frederic.cuppens}@telecom-bretagne.eu

Abstract. Many research works focus on the adoption of cloud infrastructure as a service (IaaS), where virtual machines (VM) are deployed on multiple cloud service providers (CSP). In terms of virtual resource allocation driven by security requirements, most of proposals take the aspect of cloud service customer (CSC) into account but do not address such requirements from CSP. Besides, it is a shared understanding that using a formal policy model to support the expression of security requirements can drastically ease the cloud resource management and conflict resolution. To address these theoretical limitations, our work is based on a formal model that applies organization-based access control (OrBAC) policy to IaaS resource allocation. In this paper, we first integrate the attribute-based security requirements in service level agreement (SLA) contract. After transformation, the security requirements are expressed by OrBAC rules and these rules are considered together with other non-security demands during the enforcement of resource allocation. We have implemented a prototype for VM scheduling in OpenStack-based multi-cloud environment and evaluated its performance.

Keywords: Cloud security · Resource management · Security policy

1 Introduction

Today cloud computing is essentially provider-centric. An increasing number of fiercely competing CSPs operate multiple heterogeneous clouds. In terms of IaaS, each provider offers its own, feature-rich solutions for customer VMs. More significantly, in cloud IaaS, physical hardware is usually shared by multiple virtual resources for maximizing utilization and reducing cost. Unfortunately, this vision suffers from a lack of homogeneity: many cloud virtual resources can not be deployed due to deficiencies in (1) unified expression; (2) interoperability. Lack of unified expression results in vendor lock-in: services are tightly coupled with the provider and depend on its willingness to deploy them. Lack of interoperability stems from heterogeneity of services, and more importantly of

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 105–118, 2016.
DOI: 10.1007/978-3-319-33630-5_8

service-resource mapping, not compatible across providers. For better interoperability and control, cloud brokering is nowadays the rising approach towards a user-centric vision. It may be seen as a paradigm in delivering cloud resources (e.g. compute, storage, network). With the help of brokering technology, user's security needs will be necessarily considered in cloud and these security requirements can be included in SLA contract which is a legal document where the service description is formally defined, delivered, and charged.

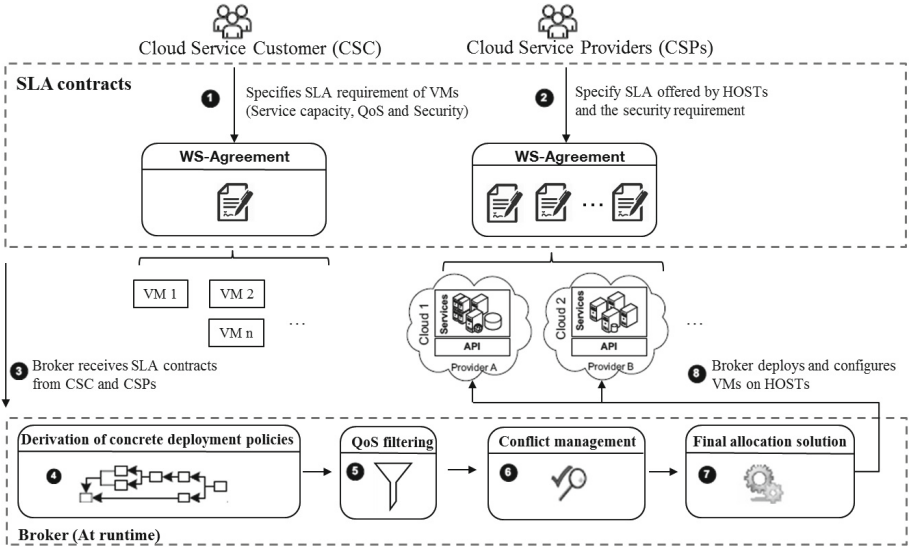


Fig. 1. The proposed policy based process to allocate virtual resources

Therefore, to overcome the aforementioned issues, we enhance the brokering technology by developing a configuration management process to allocate VMs in IaaS cloud. Shown in Fig. 1, with WS-Agreement [1] based contracts, both CSC[1] and CSP specify and manage their security requirements related to infrastructure in order to ensure end-to-end security across different components (Steps 1,2). After receiving the SLA contracts, the broker derives the concrete deployment policies according to security and non-security requirements (Steps 3,4,5). Particularly, the broker is able to arbitrate contradicting demands and make decisions (Step 6). In the end, the broker applies un algorithm to generate the final allocation solution (Step 7) then deploys and configures VMs on HOSTs (Step 8). Our method is evaluated by setting up a cloud computing environment to conduct virtual resource allocation process. Experimental results show that our approach demands minimal user (CSC and CSP)'s intervention and enables unskilled cloud users to have access to complex deployment scenarios. In particular, our solution tackles the lack of application of existing policy model that

[1] In this paper, CSC stands for the end customer of cloud.

can support security expression when dealing with multiple clouds. Our contribution meets key-functional requirement for user-centric as (i) it addresses the SLA configuration options at the IaaS layer from service capacity to security constraint. (ii) it considers multiple requirements of security and applies the OrBAC model to translate attribute-based security constraint to concrete policy. (iii) it provides conflict management to detect and handle the contradictory requirements from CSC and CSPs, with possibility to judge the policy priority by evaluating users' profiles. (iv) it proposes a resource allocation algorithm which takes resource capacity, QoS and security policy into account. To the best of our knowledge, there is no method in the literature that considers all these points.

The rest of the paper is organized as follows: Sect. 2 outlines the expression of security policy by CSC and CSPs with an exhaustive example. Section 3 illustrates the enforcement of security policy for VM allocation. Section 4 gives an implementation integrated with our solution and evaluates four experiments. Section 5 reviews existing proposals on cloud resource scheduling and security-aware allocation solution. Section 6 concludes the paper and outlines future work.

2 Expression of Security Policy

2.1 SLA Contract Expression

To generate security policies for CSC and CSP, we suggest, as a first step, to specify a generic document, which describes the requirements for service capacity, quality of service (QoS) and security constraint. SLA contract is such a document used in service negotiation and management. Based on a well-formatted template, CSP and CSC exchange their offers until reaching an agreement [2]. Among existing SLA specifications, we choose WS-Agreement because the format is open so that it can integrate various service parameters. Meanwhile, WS-Agreement is widely used by lots of research and industrial projects such as BREIN [3], IRMOS [4], and OPTIMIS [5]. Hence a WS-Agreement contract consists of name, context, service terms, guarantee terms and negotiation constraints, CSC and CSP can integrate service capacity, QoS and security requirement in its structure.

In cloud computing, the CSP's system can be viewed as a large pool of interconnected physical hosts. We use HOST to present the finite set of hosts from a CSP. Note that, VM and HOST may have multiple attributes each with their own values and these attributes can be assigned either manually by a user or automatically by the system. In terms of security requirement, as CSC and CSPs do not know the information of each other, they express their security constraints by attribute in Formulas 1, 2 and 3.

$$permission([H_{attr\_name} : H_{attr\_value}], [V_{attr\_name} : V_{attr\_value}]) \tag{1}$$

$$permission([H_{attr\_name} : H_{attr\_value}], [v_i]) \tag{2}$$

$$separate(v_i, v_j) \tag{3}$$

In the three formulas, H_{attr_name} and V_{attr_name} indicate the attribute name of HOST and VM respectively; H_{attr_value} and V_{attr_value} denote separately the attribute value of HOST and VM; each of v_i, v_j represents a unique virtual machine ID (VMID). Formulas 1 and 2 are used to specify the permission of VM allocation: HOST(s) with attributes assigned is (are) permitted to deploy VM(s). The difference is that in the first formula, the CSC specifies VM by attribute and in the second formula, VMID is given directly. These two options give the CSC more flexibility to express their security requirements. In addition, the CSC declares the coexistence constraint by Formula 3: v_i and v_j can not be allocated on the same HOST. Formula 4 is used by CSP to express the deployment prohibition. Similar with Formula 2, HOST with HOSTID h_i is not permitted to deploy VM(s) assigned with attribute.

$$prohibition([h_i], [V_{attr\_name} : V_{attr\_value}]) \tag{4}$$

In an example that we will use throughout the paper, we consider an DevOps [6] use case. DevOps is an emerged software development methodology that enhances collaboration between development, quality assurance (QA) and IT operations. Numerous companies are actively practicing DevOps since it aims to help them to maximize the predictability, efficiency, security, and maintainability of operational processes. Adoption of DevOps is being driven by many factors including using public IaaS. Suppose that a software company has to deploy 3 VMs (v1,v2,v3) in cloud for a development project. Each VM contains its metadata such as properties, required volume, QoS specification and security constraint. We suppose that each VM runs a project server and there exist three types of VM: production (prod), development (dev), and test. *prod* server runs live applications supporting the company's daily business and the data is public for e-business customers; *dev* server consists of development environment thus developers with private right can access it; *test* server is used to conduct software test between development and production phase and it is accessible by testers with private login account. At the same time, there exist 2 CSPs (h1,h2) and each has its own metadata such as price, location and state indicating if it is certificated by security audit organizations. A readable illustration of VM and HOST configuration is shown in Fig. 2.

2.2 Derivation of Security Policy

Security constraints need to be transformed to concrete security policies including VMID and HOSTID. Here we suggest using the OrBAC [7] model which supports the expression of permission and prohibition.

OrBAC in Brief. The OrBAC model is an extension of the role-based access control (RBAC) [8] model. It defines a conceptual and industrial framework to meet the needs of information security and sensitive communication and allows the policy designer to define a security policy independently. The concept of organization is fundamental in OrBAC. An organization is an active entity that

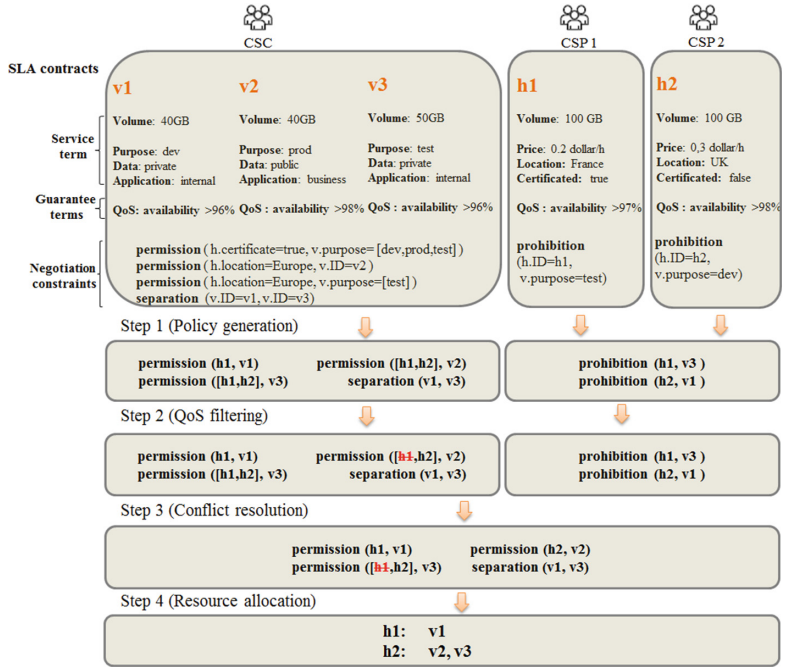


Fig. 2. An DevOps use case of virtual resource allocation

is responsible for managing a security policy. Each security policy is defined for an organization. The model is not limited to permissions, but also includes the possibility to specify prohibitions and obligations. Besides, the security rules do not apply statically but their activation may depend on contextual conditions [9]. Context [10] is defined through logical rules and it can be combined in order to express conjunctive context, disjunctive context and negative context. An OrBAC policy is defined as: **security_rule (organization, role, activity, view, context)** where **security_rule** belongs to {permission, prohibition, obligation}. Once a security policy has been specified at the organizational level, it is possible to instantiate it by assigning concrete entities to abstract entities by the predicates which assign a subject to a role, an action to an activity and an object to a view. Meanwhile, all the operations are related to a specified context:

- *empower(org, subject, role)*: in organization org, subject is empowered in role;
- *consider(org, action, activity)*: in organization org, action implements activity;
- *use(org, object, view)*: in organization org, object is used in view;
- *hold(org, subject, action, object, context)*: in organization org, subject does action on object in context.

Based on the above definitions, a concrete permission policy could be derived by the following rule[2]:

$$permission(org, role, activity, view, context)$$
$$\wedge\, empower(org, subject, role) \wedge consider(org, action, activity)$$
$$\wedge\, use(org, object, view) \wedge hold(org, subject, action, object, context)$$
$$\rightarrow Is\_Permitted(subject, action, object)$$

From Security Constraint to OrBAC Policy. Derivation of OrBAC policy from security constraint requires policy mining technology which parses the configured rules and automatically reaches an instance of high level model corresponding to the deployed policy. Most of the existing RBAC based mining methods [11,12] generate abstract policy by taking concrete rules as input. However, in our scenario, both abstract and concrete rules should be derived from attribute-based description. The following is the problem definition.

Definition 1. *Policy Mining Problem*

Given a set of attribute of Subject S (HOST), a set of attribute of Action A, a set of attributes of Objects O (VM), and SAO_attr an attribute-based subject-action-object assignment relation (Formulas 1, 2, 4), find a set of ROLES, a subject-to-role assignment SR, a set of activity ACTIVITIES, an action-to-activity assignment AA, a set of VIEWS, an object-to-view assignment OV and RAV$\subseteq$*ROLES*$\times$*ACTIVITIES*$\times$*VIEWS, a many-to-many mapping of role-to-activity-to-view assignment relation*[3]*.*

Algorithm 1 explains the generation of permission policy. First of all, after receiving contracts from CSC and CSPs, broker extracts the attribute information of each VM and HOST then generates three kinds of structures as input: (1) VM list: storing all the attributes of related VMs; (2) HOST list: storing all the attributes of related HOSTs; (3) VM security constraint list: storing all the security constraints of CSC. After initialization of policy p, concrete action *deploy* is assigned to a new *activity* (lines 2,3). Then the relevant HOSTID list ID_h_list and relevant VMID list VM_v_list are generated from each term in VM security constraint list c_v (line 4–6). For example, the relevant HOSTID and VMID for the security constraint $permission([\text{“}certificate\text{”} : \text{“}true\text{”}], [\text{“}purpose\text{”} : \text{“}dev\text{”}])$ are HOST1 and VM1. After finding the relevant VMID(s) and HOSTID(s), an abstract permission with a new role *currentRole* and new view *currentView* is created (line 7–9). Finally, all the HOSTIDs in ID_h_list are assigned to *currentRole* and all the VMIDs in VM_v_list are assigned to *currentView* (line 10–15). The prohibition policy for CSP is generated in the same way by taking

[2] A concrete prohibition policy $Is\_Prohibited(subject, action, object)$ could be derived by the same way from $prohibition(org, role, activity, view, context)$.

[3] In this paper, all the rules share the same action (“deploy”), organization (“superCloud”) and context (“default”). For reasons of simplicity, we do not illustrate organization and context in algorithm and policy.

input of VM list, HOST list and HOST security constraint list. Step 1 in Fig. 2 demonstrates an example of permission and prohibition generation.

Algorithm 1. *permissionGeneration*(l_v, l_h, c_v): permission policy generation
Input: VM list l_v, HOST list l_h, VM security constraint list c_v
Output: OrBAC policy p

```
Initiate p
p.activity ← create new activity
p.consider("deploy", p.activity)
for c_vi in c_v do
    ID_h_list ← get relevant HOSTID(s) from l_h
    ID_v_list ← get relevant VMID(s) from l_v
    p.currentRole ← create new role for HOSTs in ID_h_list
    p.currentView ← create new view for VMs in ID_v_list
    p_i ← create permission: permission(p.currentRole, p.activiy, p.currentView)
    for ID_hi in ID_h_list do
        p.empower(ID_hi, p.currentRole)
    end for
    for ID_vi in ID_v_list do
        p.use(ID_vi, p.currentView)
    end for
end for
return p
```

3 Enforcement of Security Policy

3.1 QoS Filtering

Shown in Step 2 of Fig. 2, this process aims to disable the permission which does not satisfy the QoS constraint. To this end, an evaluation between VM's performance requirements and HOST's capacity will be conducted. For example, in our scenario, QoS requirements contain the term of availability and the deployment permission between VM2 and HOST1 is disabled.

3.2 Conflict Management

After generating OrBAC policies from security constraint and executing QoS filtering, the broker aggregates permission rules of CSC and prohibition rules of CSP like:

$$permission(\{h_i\}, v_k) \tag{5}$$

$$prohibition(h_j, \{v_l\}) \tag{6}$$

In Formula 5, each VM v_k has a set of hosts $\{h_i\}$ which allow it to be deployed and in Formula 6, a set of VM $\{v_l\}$ are not permitted to deploy on the HOST

h_j. The rewriting of rules is used to detect conflicts between permissions and prohibitions. A conflict corresponds to a situation where a subject HOST is both permitted and prohibited to perform a given action *deploy* on a given object VM. We divide conflicts into the following two types and for each type an allocation solution is proposed.

Type I: Conflict With Concession Space. Defined in Formula 7, HOST h_j is permitted and prohibited simultaneously to deploy VM v_k. In fact, except for h_j, VM v_k has other allocation solutions. In this case, we disable h_j from the allocation permissions of v_k (Formula 8). For example, in step 3 of Fig. 2, $permission(\{h_1, h_2\}, v_3)$ and $prohibition(h_1, v_3)$ belong to this type and the solution is disabling $permission(h_1, v_3)$.

$$\begin{aligned} conflict\_TypeI(h_j, v_k) \leftarrow permission(\{h_i\}, v_k) \ \wedge \ prohibition(h_j, \{v_l\}) \\ \wedge \ h_j \in \{h_i\} \ \wedge \ v_k \in \{v_l\} \ \wedge \ (\{h_i\} \setminus h_j) \neq \phi \end{aligned} \tag{7}$$

$$disable(permission(h_j, v_k)) \leftarrow conflict\_TypeI(h_j, v_k) \tag{8}$$

Type II: Conflict Without Concession Space. Shown in Formula 9, compared with the conflict of type I, the difference is that in Type II, except for h_i, VM v_k has no other deployment solutions. In this case, we adopt a priority based approach proposed in [13] and introduce two labels $p(h)$ and $p(v)$ as priorities of VM and HOST. $p_1 \prec p_2$ means that p_2 has higher priority than p_1. As virtual resource allocation is related to different factors such as risk and trust, the priorities could be predefined by users or determined by the broker. For example, some of CSPs' prohibitions can be disabled by the broker in case that the CSC has a low risk score. Making decisions on priority is beyond the scope of this paper and here we suppose that CSPs obtain higher priority to fulfill all their security requirements. Thus, in Formula 10, the current conflict resolution is disabling the permission of h_i. For example, the solution for $permission(h_3, v_1)$ and $prohibition(h_3, v_1)$ is disabling the former rule.

$$\begin{aligned} conflict\_TypeII(h_i, v_k) \leftarrow permission(h_i, v_k) \ \wedge \ prohibition(h_j, \{v_l\}) \\ \wedge \ h_i = h_j \ \wedge \ v_k \in \{v_l\} \end{aligned} \tag{9}$$

$$disable(permission(h_i, v_k)) \leftarrow conflict\_TypeII(h_i, v_k) \wedge \ p(v_k) \prec p(h_i) \tag{10}$$

3.3 Virtual Resource Allocation

The aim of previous steps is to generate the final VM allocation solution. Without loss of generality, we demonstrate the generation of allocation solution from security policy by considering the CSC's preference on price. Algorithm 2 shows the resource allocation process. It takes permission policy p, VM list l_v, HOST list l_h and separation constraint c as input and generates the deployment solution which maps VMs to HOSTs. In each permission rule, VMID and a list of its

possible target HOSTs are extracted (line 1–4). To satisfy the price preference of CSC, the target HOSTs are ranked from low price to high price (line 5) thus the one with the lower price will be chosen preferentially. The final deployment solution depends on mainly two factors (line 9): (1) if the VM has coexistence conflict with the VMs which have been already deployed on the HOST. (2) if the HOST has enough volume to deploy the VM. Step 4 in Fig. 2 shows an example of resource allocation.

Algorithm 2. *resourceAllocation*(p, l_v, l_h, c): virtual machine allocation
Input: OrBAC permission p, VM list l_v, HOST list l_h, separation constraint c
Output: deployment solution

```
1: for each concrete rule r_i in p do
2:   if r_i is active then
3:     ID_vi ← get object in r_i
4:     ID_h_list ← get all the HOSTIDs permitted for ID_vi in r_i
5:     Rank ID_h_list from low price to high price
6:     for ID_hj in ID_h_list do
7:       v_i ← get VM from l_v by ID_vi
8:       h_j ← get HOST from l_h by ID_hj
9:       if ID_vi not in separation constraint c
         and h_j has enough volume for v_i
         and v_i has not been allocated then
10:        add (v_i attaches host h_j) to solution
11:      end if
12:    end for
13:  end if
14: end for
15: return solution
```

4 Implementation and Evaluation

SUPERCLOUD [14] is a European project which aims to support user-centric deployments across multi-cloud and enable the composition of innovative trustworthy services. Its main objective is to build a security management architecture and infrastructure to fulfill the vision of user-centric secure and dependable clouds of clouds. One use case is developing a middle-ware layer between CSC and CSPs and this middle-ware could allocate virtual resources on physical infrastructures. In this context, there is a need to consider a multi-cloud environment with security constraints. For example, virtual resources should not be mapped to physical resources that do not comply with its security requirements; physical resources should not deploy virtual resources that are potentially harmful to its operation; or virtual resources should not coexist on the same physical resource as another potentially malicious virtual resource [15].

In order to implement and evaluate our virtual resource allocation framework, we setup an IaaS cloud environment on a physical machine (Intel(R) Core(TM) i7-4600U 2.7 GHz with 16 GB of RAM running Windows 7). Then different VMs (2 cores and 2 GB of RAM) are created on VirtualBox platform with Ubuntu system. We now install DevStack [16] based cloud framework, a quick installation of OpenStack [17] ideal for experimentation. Each VM is regarded as a physical HOST for the purpose of experimentation. At the same time, a JAVA based program runs as cloud broker and it connects VirtualBox platform by SSH protocol. The OrBAC policy is generated and managed by the JAVA-based OrBAC API [18]. Figure 3 illustrates our experimental architecture.

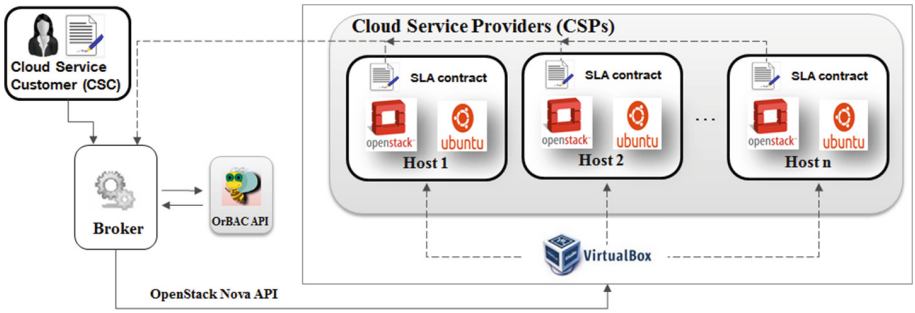


Fig. 3. Implementation for virtual resource allocation

4.1 Experiment 1: Contract Processing

This experiment measures the duration for contract processing which is the runtime required by the broker to process the JSON [19] based WS-Agreement file and generates VM and HOST list. Since there does not exist a great difference between SLA contracts of VM and HOST, here we measure contract processing time for VMs. We vary the VM number from 0 to 125 and for each number we randomly generate service attributes in different quantity from 5 to 20. Figure 4 shows the result. For a small scope of VM and attribute number, the runtime is very low (30 ms). The time increases with bigger scope of VM and attribute number. The maximum duration of the experiment is less than 100ms which indicates that the runtime is acceptable.

4.2 Experiment 2: Policy Generation

In the second experiment, we analyze the required time for OrBAC policy generation (Algorithm 1 for permission and similar algorithm for prohibition generation) once contracts are processed by the broker. In Fig. 5, we study the amount of time the broker takes to generate security policies with increasing number of

VM and HOST. For example, 60 as values in x-axis and y-axis indicates that there exist 60 VMs and HOSTs and the corresponding value in z-axis (400 ms) shows the short time needed to generate the OrBAC policies.

4.3 Experiment 3: Allocation Latency

Our third experiment investigates the impact of VM number and HOST number on the execution time of Algorithm 2. In Fig. 6, VM and HOST number vary from 10 to 60. Given 60 as VM and HOST number, the allocation latency takes about only 1 s. In real case, as HOST number is limited, the estimation of allocation latency is acceptable and it confirms the efficiency of our resource allocation algorithm.

4.4 Experiment 4: Price

The experiment measures the cost for CSC after VM allocation. We generate VMs randomly from 10 to 60 and configure 8 HOSTs. For simplicity, each HOST is supposed to provide only one type of IaaS solution with price fixed from 0.02 dollars/hour to 0.08 dollars/hour[4]. Then we compare the total price between two allocation solutions (Fig. 7). The first solution is Algorithm 2 which concerns CSC's price preference and the second solution does not consider it thus VMs are allocated arbitrary on HOSTs. As a result, Algorithm 2 shows a great advantage in reducing the deployment cost.

5 Related Work

Although virtual resource scheduling problems are NP-complete, it is well-studied by the research community by proposing various heuristic and approximate approaches for addressing different issues. Among three service models (SaaS, PaaS and IaaS) of cloud computing, virtual resource allocation in IaaS cloud has been considered by some works in the literature. Some of these works [20,21] focus on the capacity of CSP. In this case, some strategies like immediate, best effort and Nash equilibrium [22] have been applied to allocation algorithm in order to optimize the deployment algorithm with constraints such as QoS and energy [23]. Another effort is SLA-oriented resource management [24]. Among lots of requirements of CSC, security is a critical issue to be taken into account [25]. Bernsmed et al. [26] present a security SLA framework for cloud computing to help potential CSCs to identify necessary protection mechanisms and facilitate automatic service composition. Berger et al. [27]

[4] The prices are inspired from current cloud IaaS solution of Amazon EC2 and Microsoft Azure. For example, in Amazon EC2, price for the instance of m4.xlarge (4 cores, 16 GB RAM) is 0.239$/h and it costs 0.308$/h (4 cores, 7 GB RAM) for the instance of A3 in Microsoft Azure.

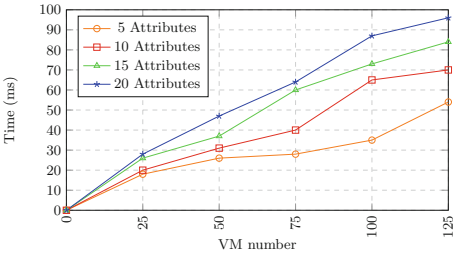


Fig. 4. Time for contract processing

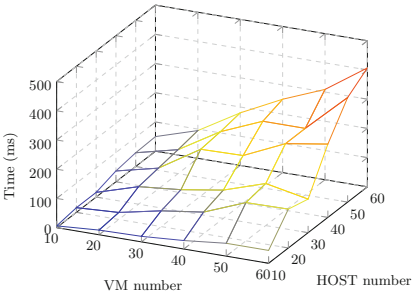


Fig. 5. Time for policy generation

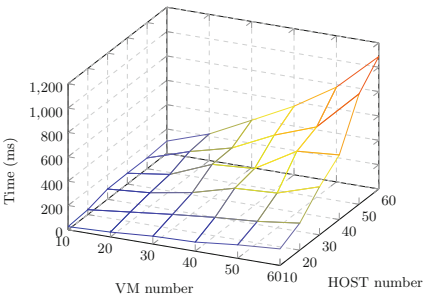


Fig. 6. Latency for VM allocation

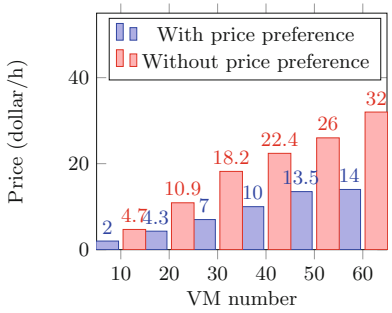


Fig. 7. Total price for VM allocation

take isolation constraint and integrity guarantee into consideration and implement controlled access to network storage based on security labels. In [28], different virtual resource orchestration constraints are resumed and expressed by attribute-based paradigm. Regarding these constraints, a conflict-free strategy is developed to mitigate risks in IaaS Cloud [29]. Most of above works have been motivated from security requirements expressed by CSC. In [30], CSP specifies its security requirements including forbid constraint which forbids a set of VM instances from being allocated on a specified HOST. However, in multi-cloud environment, as CSC and CSPs do not have vision of each other before establishing contract, specifying security requirement can be very tricky for both sides. The main focus of these efforts is scheduling VMs either for the purpose of high-performance computing or satisfying security constraints according to the requirements of CSC. Our approach is to capture security and non-security requirement from both CSC and CSP, and apply a formal policy model to drive virtual resource allocation.

6 Conclusion

In this paper, we have presented, formalized and enforced security requirement for virtual resource allocation. We first present the SLA contracts for CSC and

CSPs which contain service capacity, QoS and security constraint. We then transform the attribute-based SLA contract to concrete OrBAC policies. Finally, we allocate virtual resources after resolving conflicts in policies and demonstrate the efficiency and reliability of our solution by OpenStack-based implementation.

In future works, we plan to investigate the decision making during the conflict resolution. Another potential direction is to develop a suitable front-end application interface for SLA contract specification.

Acknowledgments. The work reported in this paper has been supported by ANRT (Association Nationale de la Recherche et de la Technologie) and Orange as a CIFRE (Conventions Industrielles de Formation par la REcherche) thesis and the work of Nora Cuppens-Boulahia and Frédéric Cuppens has been partially carried out in the SUPERCLOUD project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 643964.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (ws-agreement). In: Open Grid Forum. vol. 128, 216 (2007)
2. Li, Y., Cuppens-Boulahia, N., Crom, J.M., Cuppens, F., Frey, V.: Reaching agreement in security policy negotiation. In: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 98–105. IEEE (2014)
3. Muñoz, H., Kotsiopoulos, I., Micsik, A., Koller, B., Mora, J.: Flexible SLA negotiation using semantic annotations. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 165–175. Springer, Heidelberg (2010)
4. http://www.irmosproject.eu/
5. Ziegler, W., Jiang, M., Konstanteli, K.: Optimis sla framework and term languages for slas in cloud environment. OPTIMIS Project Deliverable D 2 (2011)
6. http://en.wikipedia.org/wiki/DevOps
7. Kalam, A.A.E., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks, POLICY 2003, pp. 120–131. IEEE (2003)
8. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **2**, 38–47 (1996)
9. Cuppens, F., Cuppens-Boulahia, N.: Modeling contextual security policies. Int. J. Inf. Secur. **7**(4), 285–305 (2008)
10. Coma, C., Cuppens-Boulahia, N., Cuppens, F., Cavalli, A.R.: Context ontology for secure interoperability. In: Third International Conference on Availability, Reliability and Security, ARES 2008, pp. 821–827. IEEE (2008)
11. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, pp. 175–184. ACM (2007)
12. Hachana, S., Cuppens-Boulahia, N., Cuppens, F.: Mining a high level access control policy in a network with multiple firewalls. J. Inf. Secur. Appl. **20**, 61–73 (2015)

13. Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High level conflict management strategies in advanced access control models. Electron. Notes Theor. Comput. Sci. **186**, 3–26 (2007)
14. http://www.supercloud-project.eu/
15. Fernando, M.V., Ramos, N.N.: Preliminary architecture of the multi-cloud network virtualization infrastructure. Technical report (2015)
16. http://docs.openstack.org/developer/devstack/
17. https://www.openstack.org/
18. Autrel, F., Cuppens, F., Cuppens-Boulahia, N., Coma, C.: Motorbac 2: a security policy tool. In: 3rd Conference on Security in Network Architectures and Information Systems (SAR-SSI 2008), pp. 273–288. Loctudy, France (2008)
19. http://en.wikipedia.org/wiki/JSON
20. Ferreira Leite, A., Alves, V., Nunes Rodrigues, G., Tadonki, C., Eisenbeis, C., Alves, M., de Melo, A.C.: Automating resource selection and configuration in inter-clouds through a software product line method. In: 2015 IEEE 8th International Conference on Cloud Computing (CLOUD), pp. 726–733. IEEE (2015)
21. Nathani, A., Chaudhary, S., Somani, G.: Policy based resource allocation in IaaS cloud. Future Gener. Comput. Syst. **28**(1), 94–103 (2012)
22. Wei, G., Vasilakos, A.V., Zheng, Y., Xiong, N.: A game-theoretic method of fair resource allocation for cloud computing services. J Supercomputing **54**(2), 252–269 (2010)
23. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**(5), 755–768 (2012)
24. Buyya, R., Garg, S.K., Calheiros, R.N.: Sla-oriented resource provisioning for cloud computing: challenges, architecture, and solutions. In: International Conference on Cloud and Service Computing (CSC 2011), pp. 1–10. IEEE (2011)
25. Li, Y., Cuppens-Boulahia, N., Crom, J.-M., Cuppens, F., Frey, V., Ji, X.: Similarity measure for security policies in service provider selection. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2015. LNCS, vol. 9478, pp. 227–242. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26961-0_14
26. Bernsmed, K., Jaatun, M.G., Undheim, A.: Security in service level agreements for cloud computing. In: CLOSER, pp. 636–642 (2011)
27. Berger, S., Cáceres, R., Goldman, K., Pendarakis, D., Perez, R., Rao, J.R., Rom, E., Sailer, R., Schildhauer, W., Srinivasan, D., et al.: Security for the cloud infrastructure: trusted virtual data center implementation. IBM J. Res. Dev. **53**(4), 1–12 (2009)
28. Bijon, K., Krishnan, R., Sandhu, R.: Virtual resource orchestration constraints in cloud infrastructure as a service. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pp. 183–194. ACM (2015)
29. Bijon, K., Krishnan, R., Sandhu, R.: Mitigating multi-tenancy risks in iaas cloud through constraints-driven virtual resource scheduling. In: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, pp. 63–74. ACM (2015)
30. Jhawar, R., Piuri, V., Samarati, P.: Supporting security requirements for resource management in cloud computing. In: IEEE 15th International Conference on Computational Science and Engineering (CSE 2012), pp. 170–177. IEEE (2012)

Software Defined Networking Reactive Stateful Firewall

Salaheddine Zerkane[1(✉)], David Espes[2], Philippe Le Parc[2], and Frederic Cuppens[3]

[1] LabSTICC, IRT B<>COM, UBO, Télécom Bretagne, 35510 Cesson-Sévigné, France
Salaheddine.ZERKANE@b-com.com
[2] LabSTICC, IRT B<>COM, UBO, 29200 Brest, France
{David.Espes,Philippe.Le-Parc}@univ-brest.fr
[3] LabSTICC, IRT B<>COM, Télécom Bretagne, 35510 Cesson-Sévigné, France
Frederic.Cuppens@telecom-bretagne.eu

Abstract. Network security is a crucial issue of Software Defined Networking (SDN). It is probably, one of the key features for the success and the future pervasion of the SDN technology. In this perspective, we propose a SDN reactive stateful firewall. Our solution is integrated into the SDN architecture. The application filters TCP communications according to the network security policies. It records and processes the different states of connections and interprets their possible transitions into OpenFlow (OF) rules. The proposition uses a reactive behavior in order to reduce the number of OpenFlow rules in the data plane devices and to mitigate some Denial of Service (DoS) attacks like SYN Flooding. The firewall processes the Finite State Machine of network protocols so as to withdraw useless traffic not corresponding to their transitions' conditions.

In terms of cost efficiency, our proposal empowers the behavior of Openflow compatible devices to make them behaving like stateful firewalls. Therefore, organizations do not need to spend money and resources on buying and maintaining conventional firewalls. Furthermore, we propose an orchestrator in order to spread and to reinforce security policies in the whole network with a fine grained strategy. It is thereupon able to secure the network by filtering the traffic related to an application, a node, a subnetwork connected to a data plane device, a sub SDN network connected to a controller, traffic between different links, etc. The deployment of rules of the firewall becomes flexible according to a holistic network view provided by the management plane. In addition, the solution enlarges the security perimeter inside the network by securing accesses between its internal nodes.

Keywords: Software defined networking · Stateful firewall · Security · Orchestration · TCP

1 Introduction

Classical networks are complex due to the lack of abstraction and due to the heterogeneity of the network infrastructure. They are costly in terms of deployment, maintenance and reconfiguration. Also, their structure is statically defined which makes

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 119–132, 2016.
DOI: 10.1007/978-3-319-33630-5_9

tedious their provisioning and upgrading. In this context, Software Defined Networking (SDN) proposes new network architecture [1] to face the challenges of legacy networks. It is based on the physical separation of the data plane and the control plane.

The SDN architecture is organized in two layers. The data plane is responsible for forwarding the network traffic. It is organized into a set of SDN compatible devices connected to each other. The control plane embeds the network intelligence: the controller and the network applications. It is responsible for network configuration and for programming the data plane devices. It offers also an interface to the network applications, to enable them manipulating the data plane layer. They interact with the controller by a Northbound API which allows them also to collect network data and to transfer their commands to the controller, via a specific interface.

The controller interacts with the data plane via a standardized southbound API. OpenFlow [2, 3] is the most common interface. It enables the controller to install Openflow rules in the data plane layer and reprogram it through its flow tables. A flow table is a collection of flow entries. Each entry is a composition of matching fields, an instruction describing the way of executing a set of actions and many counters to keep traffic statistics. The data plane devices process the traffic according to their OpenFlow tables. The inward packets' headers are compared to the matching fields. If there is a correspondence between them then the instruction is executed. More, the controller can add, modify and delete flow entries. It collects the counters and may receive encapsulated packets in Openflow format (packet-in) from the data plane devices to process them.

Potentially, SDN will offer [4] advanced abstractions by adding visibility to network applications and services and by simplifying network administration. It will enable transparent levels of scalability while elevating user experience. It will save costs of network provisioning, deployment and maintenance. Additionally, it will enhance network agility by easing network function virtualization and automating network configuration.

SDN security is challenging and two sided [5]. On one side, SDN facilitates the development and the integration of flexible, efficient and controllable security solutions. It empowers security applications by providing them a network holistic view. However, on the other side, it introduces new vulnerabilities into the network. Some of them can have major impacts on the network. For example, breaching the controller will put the entire network beneath the attacker's control.

We propose a SDN stateful reactive firewall to protect the network from illicit access. SDN firewalls offer many advantages compared to traditional firewalls. They are cost effective because they enable to elevate the data plane with firewalling behavior. Thus, legacy firewall devices are no longer needed. They are also flexible since the controller can at any time reconfigure them and deploy them in any place. They offer a management interface for administrators to ease their tasks. Also, they enable them to apply efficiently the network security policies in the data plane devices.

Many stateless SDN firewall had been proposed in the SDN realm. We are the first to propose an operational stateful SDN firewall. Moreover, our solution can also handle stateless communications.

The proposed firewall behaves in a reactive mode according to a generic algorithm. The later takes in entry the Finite State Machine (FSM) of any network protocol and produces the appropriate Firewall machine. In each transition, it incorporates as set of

OF rules to express the corresponding action. We propose a first implementation to process TCP traffic. It receives connection synchronization packets, verifies their legitimacy against the security policies and validates them. The reactive behavior of the firewall saves flow table's space in the data plane devices by reducing the number of the installed flow rules. Besides, the firewall processes the traffic according to the states of the connection. For each connection's state, it receives only the traffic corresponding to the transitions from this state. This mechanism enables to restrict the traffic to useful communications and to mitigate some DoS attacks like Syn flooding.

Our solution is entirely integrated into the SDN architecture. We take full advantage of the SDN paradigm in terms of automatization, flexibility, abstraction and efficiency. In this regards, the firewall spreads dynamically the security policies according to its global view and adapts its behavior whenever the topology is updated. It installs its rules in any OpenFlow compatible device and enables the later to behave according to the access control decisions alike a firewall. Besides, it enables the user to express its policies without worrying about their installation and maintenance in the network. In addition, it enables to save costs related to repetitive firewall maintenance and provisioning tasks.

The architecture of the solution is as following. The application layer runs above the controller. It expresses the logic of the firewall. Below, in the data plane layer, we integrate a set of Openflow rules. These rules express the security policies according to OpenFlow. Besides the two conventional SDN layers, we propose a management level to orchestrate and reinforce the security policies. It enables the configuration of the firewall management and provides the administrator with a global view on the network.

The remainder of this paper is organized as follow. In Sect. 2, we describe the state of the art of SDN firewalls. In Sect. 3, we present the architecture and algorithm of our solution. We provide in Sect. 4 the details of its implementation and the results of the performance tests. Finally, we conclude with some insights and related perspectives, in Sect. 5.

2 Related Work

A firewall is a mechanism used to protect a network by filtering the traffic coming or going to an untrusted network [6]. It matches the packets' headers of the untrusted network with a set of security policies, and it filters them in order to allow only the accepted traffic to enter the network. A security policy is a set of filtering rules expressing the security policies of the organization [7]. Each filtering rule gathers 3 blocks. (1) A priority is used to determine the order of the rule's execution. (2) Many matching fields enable the classification of a packet based on the values of its headers. (3) An action is applied to allow or deny the packet to its destination. There are mainly 3 types of firewalls [8–10]: stateless, stateful and application firewalls.

Stateless firewalls neither process nor keep in memory the different states of a connection. They do not take into consideration dynamic network information such as port source negotiation. Therefore, they are vulnerable and can be breached. Stateful firewalls have been introduced to resolve the shortcomings of the previous technology. They record in their memory the different states of a connection. They use, in addition,

the attributes related to the states of a connection in their matching fields. Application firewalls are advanced stateful firewalls. They use application layer matching fields to classify packets and handle application level threats.

There are several works in the field of SDN stateless firewalls. Most of these solutions use Openflow rules to express the firewall security policies. The authors in [11–13] propose such SDN stateless firewalls. Their solutions forward to the controller the unknown traffic for processing. The controller then, parses the packet headers and it matches their values with the policy rules. The administrator can install the firewall policy rules in the data plane using the Openflow protocol. In this case the controller interprets these policies into Openflow rules and sends them to the data plane devices.

Besides, many SDN controllers propose their own version of stateless SDN firewall [14]. These firewalls lack of user graphical interface and are connectionless. FleXam [15–17] is an extension of Openflow which integrates a stateless firewall. It runs on the controller and provides a means to specify a set of flow filters on specified parts of a packet. Then, it applies the associated action to the packets.

Moreover, some SDN frameworks have been proposed to implement stateless firewall functions. FRESCO [18, 19] offers the possibility to instantiate predefined security modules and connect them together into a SDN stateless firewall. Flowguard [20, 21] is another SDN framework. It provides means to build Openflow stateless firewall rules into the data plane and to verify flow rule policy violations.

We have found in the literature one proposition [22] related to SDN stateful firewalls. The solution is based on Openflow and adds three new tables. This firewall keeps a table in the controller to save the connections' states and to synchronize the controller with the connection updates happened on the data plane tables. The other two tables are in the switch. One table manages the actual states of the connections and the other enables the data plane devices to process the next states. The limitation of this firewall relies on its excessive memory space consumption in the data plane and the volume of the generated traffic with the controller in order to keep it synchronized.

3 Firewall Design

Our solution is integrated into the SDN architecture and uses Openflow as a way to express the security policies. It is stateful since it records the states of the connections, and processes the information related to these states to protect the network. The behavior of the firewall is reactive. It reacts to the traffic by filtering packets and accordingly installs the appropriate Openflow rules to manage their connection.

3.1 Firewall General Architecture

Our solution (see Fig. 1) is distributed into 3 levels. (1) The higher level offers orchestration services including a security policies management. (2) A stateful firewall application is integrated on the top of the control layer. It is responsible for processing the states of the connections and installing the Openflow rules and filters connection requests according to the security rules. (3) An OpenFlow level expresses the security policies with OF rules which are installed in the data plane layer.

The orchestrator offers a management interface so that the administrator can express the security policies and access to the network global view. The orchestrator can be considered as a federation point because it collects the security policies and propagates them to the controllers. Also, it collects network data such as statistics and network logs and keeps them into its database. Based on these data, the orchestrator constructs a holistic network view including the network topology. In order to reinforce and propagate the security policies, it uses an Access Table. It contains all the stateful and stateless security policies specified by the administrator. The orchestrator manages this table to propagate the security policies into the network.

The orchestrator can also configure the behavior of the Firewall Applications dynamically. When the latter receives a new configuration, it loads the corresponding module and stops the old one. Such configurations options are the behavior mode: stateful or stateless, the event mode: periodic (according to a timer) or instantaneous (according to a sensor) and the topology discovering mode: static (topology data provided by the user) or dynamic (by learning dynamically the topology).

Each time a new controller joins the orchestrator, the latter sends to this controller the security policies that concern the part of the network it controls. These security policies are recorded in the Access table of the firewall application. Each controller is connected to the orchestrator by a Rest API. It enables any type of controller to interact with our orchestrator. It ensures also the interactions between each controller and its instance of the Firewall Application.

When a new data plane device joins the controller, the firewall application produces an OF universal rule and sends it to the new connected side. This rule matches with connection initialization packets and executes a forward to controller action. The firewall application also configures the data plane device by setting its table miss entry. In this case, all packets without a correspondence are dropped by default. Hence, each synchronization packet is sent to the controller which then transmits it to the firewall application. The latter then, verifies if the connection is legitimate or not using the Access table. In case the connection is rejected the packet is dropped.

Each instance of the firewall application uses a state table to record the connections' states and their attributes. This table enables the application to keep track of the connection, its state and its possible transitions to the next states. It is also used to create State OF rules in order to restrict the traffic only to the packets corresponding to the actual state and its possible transitions. This mechanism guaranties that the controller receives only the events triggering the transitions from the actual state of the Active connection. As an outcome, the Firewall Application reduces the load on the controller and mitigates some DoS attacks like Syn flooding. Because, we can restrict the number of synchronization requests for a connection, and clean the traffic from packets which are inconsistent with the connection state. For example, when a connection's synchronization succeeds, the firewall denies any further synchronization demand for it.

The data plane devices store in their Openflow tables (see Fig. 1) the security policies in the Openflow rule structure. The universal rule matches with any synchronization packet and executes a forward to the controller action. The *Stateless rules* express the stateless policies of the firewall. The *Stateful rules* correspond to the stateful behavior of the firewall Application and the tables miss entry to process

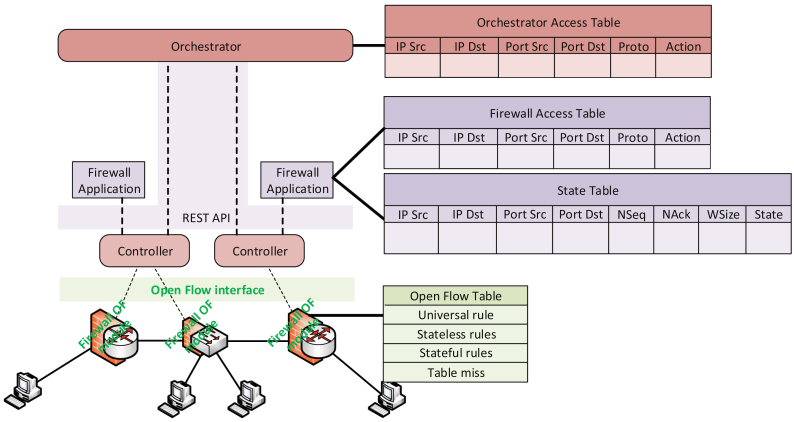


Fig. 1. SDN Stateful firewall general architecture

unmatched traffic. Except for synchronization packets, any other traffic is dropped in the data plane devices, if it is not corresponding to a legitimate connection state in the firewall application. Therefore it contributes to mitigate some DoS attacks since spoofed traffic will be directly dropped in the data plane devices.

The data plane devices perform firewalling behaviors by running the above OF entries. In the SDN architecture, each data plane device can be seen as a firewall from an external point of view. Thus, instead of using dedicated and specialized hardware, the SDN firewall elevates the behavior of the switch by reprogramming it according to the network security policies.

3.2 Firewall Generic Algorithm

The Class *Firewall_General_Behaviour* describes the generic algorithm of the firewall. The algorithm is thought in a way to process the Finite States Machine (FSM) of any communication protocol. It takes as entries the observed network events. Then, it verifies them with the preconditions of the actual FSM's state. If they fulfill the pre-conditions, it applies the corresponding actions. The firewall adapts the FSM's actions according to OF protocol by interpreting the original actions into OF rules. Then, it transits to the new state.

In our work we apply the generic algorithm to TCP communications. It has been instantiated with TCP states, transitions, their preconditions and actions. We also encapsulate some transitions' actions with Openflow rules in order to comply with Openflow standard.

In the first step, the Orchestrator sends the Universal OF Rule and the settings of the table miss to all the Firewall Application instances. The following program code describes the structure of the universal OF Rule for TCP communications.

```
def Universal_OF_Rule():
1. action = OF_ActionOutput (FORWARD_CONTROLLER)
2. instruction = Instructions(OF_APPLY_ACTIONS, action)
3. matching_fields = OF_Match(eth_type = IPv4, IP_PROTO =
   TCP, TCP_FLAGS = (SYN) )
4. OF_RULE = OF_FlowMod (match = matching_fields, command
   = ADD_RULE, priority = 1, instructions = instruction)
5. send_msg (OF_RULE)
```

Through the controller, the Firewall Application is constantly listening to a potential connection of new data plane devices. It reacts to the network events by propagating the received Universal Rule and the settings of the table miss. It also, installs the Openflow Stateful Rules when the state transitions are triggered. The Firewall Application observes networks events according to two modes:

1. **Periodic Mode:** in this mode (see the *Firewall_Periodic_Mode Class*) the firewall sets a timer to observe periodically new network events coming from the controller. When the timer reaches its threshold, it sends a request to the controller to check if any new data plane device has been connected or if any update happened in any known data plane device. Once an alteration is observed, the firewall generates the corresponding Openflow Universal Rule and the configuration of the table miss. Then, it installs them on the new data plane device.

```
Class Firewall_Periodic_Mode :
1. Read (threshold)
2. While (true):
3.    Sleep(start_time=0, end_time=threshold)
4.    If (new_data-plane-device is connected):
5.      Send(Universal_OF_Rule,new_data-plane-device)
6.      Set(Table-miss(Drop),new_data-plane-device)
```

2. **Instantaneous Mode:** The firewall (see the *Firewall_Instantaneous_Mode Class*) puts in place a sensor into the controller to collect new network events coming into it. When the sensor detects a new data plane device, it prompts the firewall immediately. Then, the firewall generates the corresponding Openflow Universal Rule with the settings of the table miss and installs them on the new data plane device.

```
Class Firewall_Instantaneous_Mode :
1. While (true):
2.   Sleep(Waking_Event=Sensor_Notification)
3.   If Sensor_Event == New_Data_Plane_Device_connected :
4.     Send(Universal_OF_Rule,new_data-plane-device)
5.     Set(Table-miss(Drop),new_data-plane-device)
```

The firewall application uses one of the previous modes to update the data plane devices. When it receives a synchronization packet, it verifies its legitimacy. It checks in the access table if the connection is accepted or denied. If there is no specified policy for the connection in the table, it is denied by default. If the connection is accepted, an

entry is created in the connection table. Then, the firewall verifies if the preconditions in the packet activate any of the transitions from the actual connection's state. If a transition is found, the firewall applies the corresponding actions associated with it, and then, it sends delete requests to the data plane device to remove the previous State Openflow rules. In the opposite case, the packet is dropped. Finally it updates the state in the state table and installs the new corresponding State OF rules. This mechanism lessens the load of the traffic into the controller, because the data plane devices send only the packets that can prompt the available transitions. Any traffic outside this zone is automatically dropped in the data plane device.

```
Class Firewall_General_Behaviour :
1. If (FW_Mode==Periodic):
2.  Firewall_Periodic_Mode();
3. Else:
4.  Firewall_Instantaneous_Mode();
5. While (true):
6.  C_Event=Collect(Controller_Events);
7.  If (C_Event.Type==Packet-in)
8.   Connection_Status =Check_Connection(C_Event)
9.   If (Not Connection_Status)
10.    Protocol_Data = C_Event.Protocol.Data
10.    Legitimacy = Check_Legimacy(Protocol_Data)
11.    If (Legitimacy):
12.     Create_Connection(Protocol_Data,State_Table)
13.    Else:
14.     Return;
16.  Else: #If Connection_Status == True
17.     Continue;
18.  Preconditions=Check_Preconditions(Protocol_Data)
19.  If (Not C_Preconditions.Status):
20.    Return;
21.  Else: #If the preconditions allow any transition
22.   C_Transition=Select_Transitions(C_Preconditions);
23.   Apply(C_Transition.Actions);
24.   Update(Connection_OF_Rules.Previous);
25.   Install(Connection_OF_Rules.New);
26.   Set(C_Transition.State,State_Table)
27. Else: #If C_Event.Type is not Packet-in
28.   Return;
```

4 SDN Firewall Proof of Concept

We have implemented the firewall on the RYU [23] controller using the Python language. RYU is a software component SDN controller. It provides means to use multi-threading, to parse and serialize packets and to communicate with different data plane devices. We based our implementation on the instantiation of the generic algorithm for TCP. Then, we deploy a testbed to measure the performance of the SDN firewall.

4.1 Implementation

The firewall API (see Fig. 2) comprises two packages. The orchestrator package runs in the management layer. It offers a General user Interface to manage the security policies and the OF rules. It allows adding, modifying and displaying the security policies and manages the static topology information. It offers to the administrator the possibility to configure many parameters of the firewall such as event modes, behavior modes, etc. It keeps open sockets with all the Firewall Application instances to communicate with them. Through these canals, it sends management commands and collects network events.

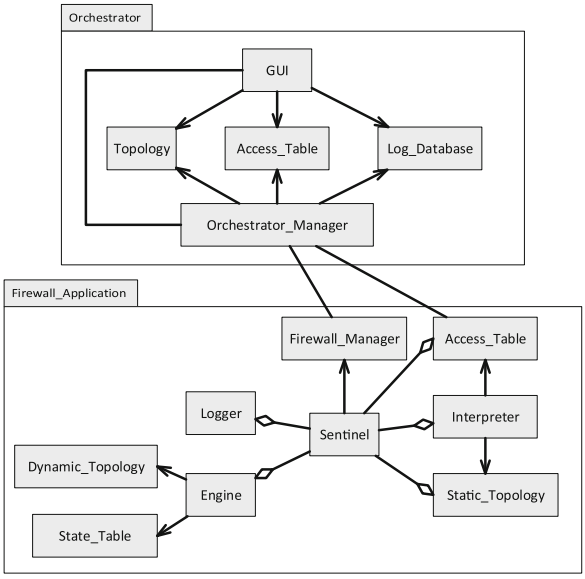


Fig. 2. Firewall API class diagram

The core package (*Firewall_Application*) is running on the Ryu Controller. It is mainly formed of the following components. (1) The *firewall Manager* Module keeps an open socket with the orchestrator. It sends the data coming from the other modules to the orchestrator and vice versa. (2) The *Interpreter* module translates the Administrator rules into Openflow rules according to the specification of the Openflow protocol. (3) The *State_Table* keeps the connection states, their properties and the firewall policies. (4) The *Logger* class collects information on the firewall components, connections data and traffic statistics. (5) The *Static_Topology* class provides data on the network topology. (6) The *Sentinel* singleton is responsible for the interaction with the controller. It configures also the firewall components and instantiates them. (7) The *Engine* class expresses the behavior of the firewall in handling all the phases of a stateful connection and in processing the communication between the client and the server.

4.2 Test and Results

We deploy and configure our test environment (see Fig. 3) in mininet [24]. The latter is a Python application to emulate virtual networks. Our mininet environment comprises 3 clients and a HTTP server. All are connected by their virtual network interfaces to a virtual switch (OVS [25]). Each client runs a Python script which generates a number of simultaneous queries to request data from the HTTP server. The Ryu controller is remotely connected to the virtual switch via the virtual channel offered by mininet. It offers an execution environment to the Firewall Application and ensures all its interactions with the virtual environment. Besides we run the orchestrator and we connect it with a socket to the firewall Application. Our environment is running under Ubuntu 14, 64 bits, 2 GB of RAM and 2 processors at 2.8 GHz. The effective average latency between the controller and the switch is 0.25 ms, while the chosen bandwidth is 1 GB/s.

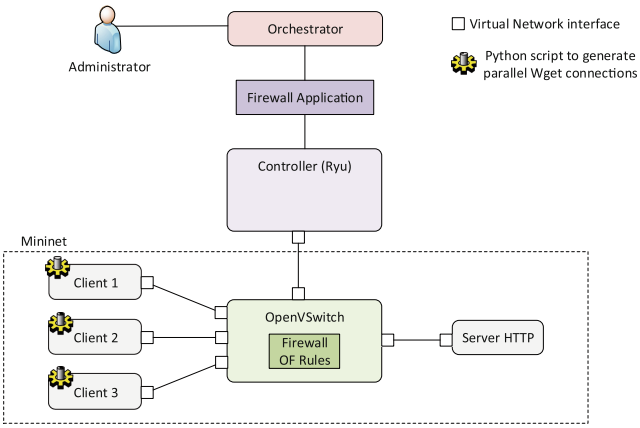


Fig. 3. SDN Firewall experiment Environment

We perform two different experiments in order to show the impact of the firewall on the connections' processing times and its effects on the user quality of service. In the first experiment (see Fig. 3) we remove from the test bed the orchestrator and the firewall Application. Then, we activate the learning switch module of the Controller so that it behaves like a learning switch. In this case, the virtual switch sends the unmatched traffic to the controller. The latter maintains dynamically a table associating each IP and Mac host addresses with an OVS port number. When the route is found in the table, the controller installs the corresponding OF Rules to enable the switch to forward directly the traffic to its destination. In case it does not find a port for the unmatched packet, it broadcasts the packet to all the switch ports and it waits for an answer to add the new correspondence (Fig. 4).

In the second experiment (see Fig. 3), we run the Firewall Application on the Controller. We connect to them the Orchestrator and we disable the learning switch module in order to let our firewall handling all the traffic.

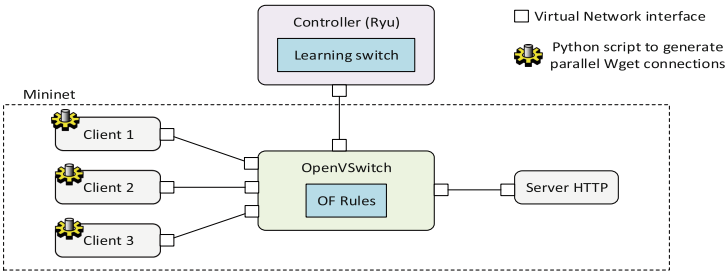


Fig. 4. SDN Learning Switch experiment Environment

In the two cases, the clients generate the same number of simultaneous TCP connections. We started the tests with 10 simultaneous connections and ended at 1000 simultaneous connections per second in a continuous and a constant interval of time. We perform in every experiment the following measurements: the average processing time of a packet-in and the average TCP connection time (the average time needed to process a complete TCP session). Furthermore, in the case of the experiment 2, we measured the maximum and minimum time of packet-in processing.

We analyze the data with two objectives. The first one is the performance of the firewall compared with a controller without a firewall (the learning switch controller). In this case, we are interested in observing how much extra time the Firewall needs to process the packet-in and the connections. Performance results are presented in Figs. 5 and 6. In the second case, we focus on the scalability of the firewall by observing the evolution of the packet-in processing time zones with the number of simultaneous connections. The results are presented in Fig. 7.

Figure 5 displays the Average packet-in processing time. We observe in both experiments a rather constant average time. The average packet-in processing time of the Firewall stays between 0.9 ms and 0.7 ms. For the learning switch controller it is almost constant around 0.5 ms. The firewall takes 0.3 ms more than the learning switch from 10 to 250 simultaneous connections then this extra time decreases to 0.2 ms till 1000 simultaneous connections. The time added by the firewall can be considered as inconsequential. It does not also inflate with the surge of the number of parallel connections.

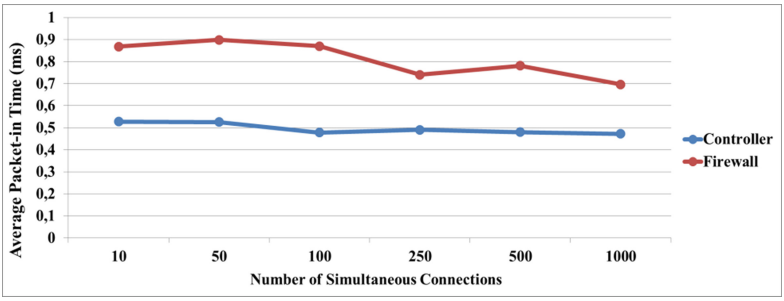


Fig. 5. Average processing times of packet-in

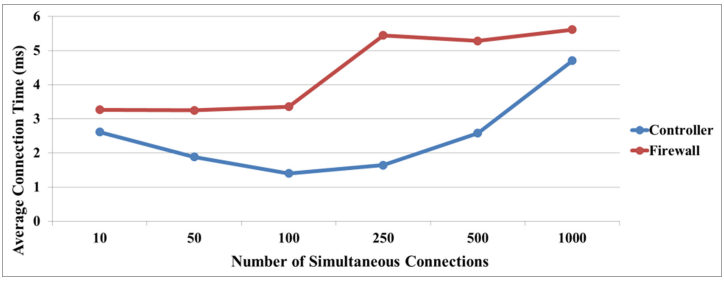


Fig. 6. Average TCP connections processing times

The results regarding the average TCP connections time are shown in Fig. 6. In the case of the firewall the Average time is almost steady (around 3.3 ms) from 10 to 100 simultaneous connections while for the learning switch the average time decreases from 2.6 ms to 1.4 ms. From 100 simultaneous connections, the average time of the firewall increases till 250 simultaneous connections (5.4 ms) and then stays almost steady. While for the learning switch it continues in each step to increase reaching at the end a value of 4.7 ms. The processing time added by the firewall increases from 0.6 ms to 3.8 ms then decreases to 0.9 ms. The extra time added by the firewall in this case is also insignificant and scale very well with the increase of the number of simultaneous connections. In terms of Quality of Experience (QoE) this time is indiscernible for the user and does not reach the TCP timeouts values.

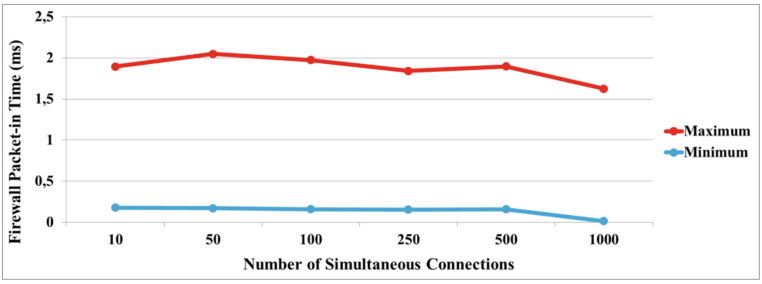


Fig. 7. Firewall Packet-in time values domain

In Fig. 7, we observe the amplitude of the Packet-in processing time values for the firewall. It is almost steady (around 1.8 ms) all along the growth of the simultaneous connections. The interval of the packet-in processing time values is as following. The maximum values are between 2 ms and 1.6 ms while the minimum values are between 0.2 ms and 0.01 ms. The maximum and minimum values are related to RYU multi-threading processing. The Engine threads are created by the Sentinel and then are put in a queue. If the queue reaches an important size, the packet-in processing time increases to the maximum values.

5 Conclusion

We introduce in this paper the first SDN reactive firewall. We speak about the advantages of our solution in terms of flexibility, performance, security enforcement and effectiveness. We discuss about its conceptual foundations based on a general algorithm specified for the TCP protocol. Finally, we show the details of its implementation, its deployment in a virtual environment and the results of the tests.

We add to the SDN architecture an orchestrator to manage the network according to a holistic view. We also integrate a Firewall Application which enforces the security policy.

In terms of performance, our solution adds a negligible delay to process packet-in or TCP connections. Regarding scalability, we show that the time processing does not increase with the number of simultaneous connections. These results are encouraging and confirm the effectiveness of our proposition.

We plan to consider the following enhancements in order to improve our solution. The first improvement will focus on the evaluation part. We will deploy the SDN test bed in a real environment. All the SDN elements will be hosted in dedicated powerful machines. We will push the firewall capabilities to their limits in order to measure the maximum number of connection that it can handle and the impacts on the network performances. In the second enhancement, we propose to develop a meta-firewall on the management plane. The orchestrator will instantiate it dynamically into different firewall applications and will place them in different SDN locations. This specialization will take into account the global view of the orchestrator, the spatial, historical and temporal context of the SDN network and the type of the network communication protocol.

References

1. Kreutz, D., Ramos, F.M.V., Verissimo, P., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: A comprehensive survey. Proc. IEEE **103**, 14–76 (2014)
2. The Open Networking Foundation, OpenFlow Switch Specification (2014)
3. Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using OpenFlow: A survey. IEEE Commun. Surv. Tutorials **16**(1), 493–512 (2014)
4. Jammal, D.M., Singh, T., Shami, A., Asal, R., Li, Y.: Software-defined networking state of the art and research challenges. J. Comput. Netw. **72**, 1–24 (2014)
5. Schehlmann, L., Abt, S., Baier, H.: Blessing or curse? Revisiting security aspects of Software-Defined Networking. In: 10th International Conference on Network and Service Management, pp. 382–387 (2014)
6. Sharma, R.K., Kalita, H.K., Issac, B.: Different firewall techniques: A survey. In: 5th ICCCNT (2014)
7. Zeidan, S., Trabelsi, Z.: A survey on firewall's early packet rejection. In: International Conference on Innovation and Information Technology, pp. 203–208 (2011)
8. Bidgoli, H.: Packet filtering and stateful firewalls, Handbook of Information Security, Threats, Vulnerabilities, Prevention, Detection, and Management, pp. 526–536. Wiley, New Jersey (2006)

9. Trabelsi, Z.: Teaching stateless and stateful firewall packet filtering: A hands-on approach. In: 16th Colloquium for Information Systems Security Education, pp. 95–102 (2012)
10. Guo, F., Chiueh, T.-C.: Traffic Analysis: From Stateful Firewall to Network Intrusion Detection System, RPE report, New York (2004)
11. Collings, J., Liu, J.: An OpenFlow-based prototype of SDN-oriented stateful hardware firewalls. In: IEEE 22nd International Conference on Network Protocols, Chapel Hill (2014)
12. Pena, J.G., Yu, W.E.: Development of a distributed firewall using software defined networking technology. In: 4th IEEE International Conference on Information Science and Technology, pp. 449–452 (2014)
13. Yoon, C., Park, T., Lee, S., Kang, H., Shin, S., Zhang, Z.: Enabling security functions with SDN: A feasibility study. Comput. Netw. **85**(1389–1286), 19–35 (2015)
14. Suh, M., Park, S.H., Lee, B., Yang, S.: Building firewall over the software-defined network controller. In: The 16th International Conference on Advanced Communications Technology, pp. 744–748 (2014)
15. Shirali-Shahreza, S., Ganjali, Y.: Efficient implementation of security applications in openflow controller with FleXam. In: 21st Annual Symposium on High-Performance Interconnects, pp. 49–54 (2013)
16. Shirali-Shahreza, S., Ganjali, Y.: FleXam: Flexible sampling extension for monitoring and security applications in OpenFlow. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 167–168 (2013)
17. Shirali-Shahreza, S., Ganjali, Y.: Empowering software defined network controller. In: IEEE International Conference on Communication, pp. 1335–1339 (2013)
18. Shin, S., Porras, P., Yegneswaran, V., Gu, G.: A framework for integrating security services into software-defined networks. In: 2013 Open Networking Summit (2013)
19. Shin, S., Porras, P., Yegneswaran, V., Fong, M., Gu, G., Tyson, M.: FRESCO: Modular composable security services for software-defined networks. In: Network and Distributed System Security Symposium, pp. 1–16 (2013)
20. Hu, H., Han, W., Ahn, G.-J., Zhao, Z.: Towards a Reliable SDN Firewall, Open Networking Summit (2014)
21. Hu, H., Han, W., Ahn, G.-J., Zhao, Z.: FLOWGUARD: Building robust firewalls for software-defined networks. In: HotSDN 2014 (2014)
22. Juan, W., Jiang, W., Shiya, C., Hongyang, J., Qianglong, K.: SDN (self-defending network) firewall state detecting method and system based on OpenFlow protocol. China Patent CN 104104561 A, 11 August 2014
23. RYU Team, component-based software defined networking framework. http://osrg.github.io/ryu/. Accessed 27 August 2015
24. Heller, B.: Reproducible network research with high-fidelity emulation. Doctoral Thesis. Stanford University (2013)
25. OpenVSwitch. http://openvswitch.org/. Accessed 2 September 2015

Phishing and Data Sharing

Teaching Phishing-Security: Which Way is Best?

Simon Stockhardt(✉), Benjamin Reinheimer(✉), Melanie Volkamer(✉), Peter Mayer, Alexandra Kunz, Philipp Rack, and Daniel Lehmann

Technische Universität Darmstadt, Darmstadt, Germany
{simon.stockhardt,benjamin.reinheimer,melanie.volkamer,peter.mayer, alexandra.kunz,philipp.rack,daniel.lehmann}@secuso.org
https://secuso.org

Abstract. Ever more processes of our daily lives are shifting into the digital realm. Consequently, users face a variety of IT-security threats with possibly severe ramifications. It has been shown that technical measures alone are insufficient to counter all threats. For instance, it takes technical measures on average 32 h before identifying and blocking phishing websites. Therefore, teaching users how to identify malicious websites is of utmost importance, if they are to be protected at all times. A number of ways to deliver the necessary knowledge to users exist. Among the most broadly used are instructor-based, computer-based and text-based training. We compare all three formats in the security context, or to be more precise in the context of anti-phishing training.

Keywords: IT-security training · User study · Computer-based training · Instructor-based training · Text-based training · Phishing

1 Introduction

As our daily lives increasingly shift into the digital world, the number and variety of security threats the average user faces on a daily basis increases as well. Technical measures are in place to mitigate these security threats, but they do not offer sufficient protection [17]. One example that demonstrates this insufficiency is phishing: it takes on average 32 h until automated phishing detection identifies and blocks malicious websites. During that time frame users will remain unprotected if not taught how to protect themselves [11,16,21]. Security training is widely accepted as one of multiple components for achieving higher end-user IT-security [2]. Corresponding training approaches exist in different formats, all offering different advantages and disadvantages. Especially instructor-based training, computer-based training, and text-based training have been proposed for delivering all kinds of IT-security knowledge to average end-users [13,14,20].

Instructor-based training describes training situations in which an instructor teaches the participants. Due to the presence of the instructor, this situation allows for real-time feedback, questions and answers, as well as shifting the focus

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 135–149, 2016.
DOI: 10.1007/978-3-319-33630-5_10

of the training to suit the learners' needs [8]. The term computer-based training describes training that is necessarily aided by technology (e.g. computers, tablets, smartphones). The learner can train individually at the most convenient times and can always stop learning to return at a later point in time. Like the instructor-based training it also allows for direct feedback on the users performance. Text-based training is based on reading material (e.g. printouts, PDF, etc.). Analogously to computer-based training, text-based training offers self-paced, individualized learning of the content. However, due to the static nature of the format text-based training does not offer the possibility for individual feedback or other interactive elements. In its basic form it does not require an instructor or electronic devices (though some forms of text delivery, e.g. through PDFs or websites, obviously necessitate a respective device). While the formats have been compared in empirical evaluations, the existing literature lacks studies comparing all three formats in the phishing context, when delivering the same content with each format.

The goal of this work is to comparatively evaluate the three formats instructor-based training, computer-based training, and text-based training when delivering the same content through each format. For this purpose we conducted a user study researching the following aspects: (1) effectiveness of transferring the knowledge to the user, (2) user satisfaction, (3) confidence, and (4) efficiency of the training formats. Our results indicate that instructor-based training transfers knowledge significantly more effectively than any other format. Instructor-based training also achieves the highest scores in user satisfaction and confidence. Furthermore, text-based training is the most efficient format (time spent with this format leads to more correct answers in comparison to the same amount of time spent with any other format).

2 Training Material

For our evaluation we use the anti-phishing training NoPhish (secuso.org/nophish), as it exists in all three different delivery formats: instructor-based, computer-based and text-based training. Also NoPhish has previously undergone research delivered as computer-based and instructor-based training and has iteratively been improved [4,5].

2.1 NoPhish Content

The NoPhish training content is based on findings from different academic disciplines. Firstly it is based on learning principles [22] such as practice, effect, repetition and direct feedback in order to deliver an effective learning experience. Secondly it is based on a user-centered design [1].

The training is split into two parts: the *introductory part* and the *main part*. The introductory part of the training material contains general information about phishing. This includes possible consequences of phishing attacks to emphasize the risks. It is based on the fact, that the URL is the only reliable

Fig. 1. Structure of the URL.

indicator when it comes to deciding whether or not a website is a phish. As shown by [10,15]. Afterwards, it explains where to find the URL, which is especially important when using a mobile device. Users are more vulnerable to phishing when using a mobilve device [9]. How the URL is structured is also explained (Fig. 1).

The main part is split into four different lessons, which cover the most common URL spoofing tricks [21]. Each lesson explains a specific spoofing trick, namely (1) *IP/Random URL*, (2) *Subdomain/Path* (e.g. https://facebook.login.com, https://login.com/facebook), (3) *Name Extension* (e.g. http://facebook-login.com) and (4) *Spelling* (e.g. http://facebok.com). Information about the spoofing trick contains detailed explanations on the type of attacks as well as a number of legitimate and fraudulent examples (Fig. 2). Examples increase in complexity during the course of the training to challenge learners. The number of examples per newly introduced spoofing trick increases with later lessons.

2.2 Training Formats

The training formats differ from each other in terms of how the training is delivered to the participants. We explain the differences in this section.

Instructor-Based Training: The exercises are given to the participants to be classified as phishing or legitimate in a plenary session. Answers are to be openly discussed in the audience. The audience is encouraged to give feedback and helpful advice if an example was answered incorrectly. During the exercises the instructor moderates the discussion and answered questions to clarify misunderstandings.

Computer-Based Training: The computer-based training format is delivered via an Android application and is self administered by the participants. The application gives direct feedback on correctness of answers. The exercise part was designed in a playful manner containing gamification elements like lives and "levels" (Fig. 3). The purpose of using gamification elements was to motivate users. Progress in the game is granted only if a predefined number of phishing and legitimate URLs has been identified correctly.

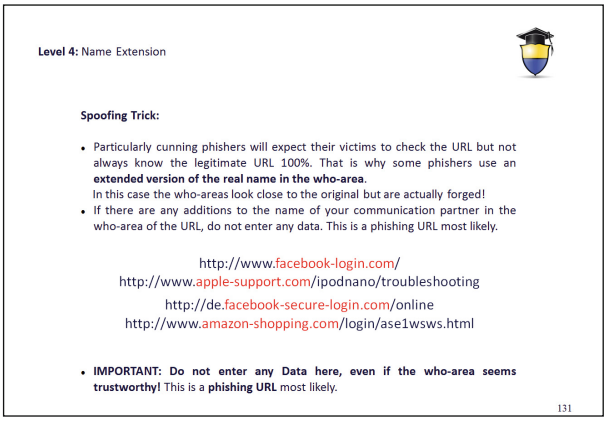


Fig. 2. Example taken from NoPhish PDF.

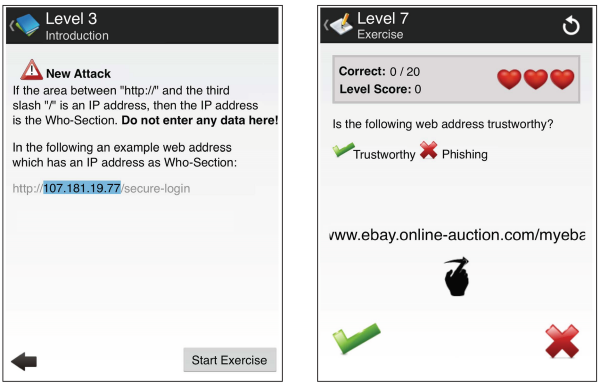


Fig. 3. Example screenshots taken from NoPhish android application.

Text-Based Training: The text-based training issues participants with the same NoPhish PDF used in instructor-based training. This training format precludes provision of feedback. Participants can take as much time as they wanted in reading through the Material.

3 Methodology

We conducted a user study to answer the following research questions:

Research Question 1 - Effectiveness: (a) How effective are the formats in transferring knowledge to the participants? (b) Are there significant differences between the three formats?

Research Question 2 - User satisfaction: (a) How satisfying is it to learn with each format? (b) Are there significant differences between the three formats?

Research Question 3 - Confidence: (a) What impact do the training formats have on people's confidence regarding their own abilities in correctly identifying legitimate and fraudulent websites attacks? (b) Are there significant differences between the three formats?

Research Question 4 - Time efficiency: (a) How much does a training group increase in correctly recognizing phishing and legitimate URLs per minute spent with the training? (b) Are there significant differences between the three formats?

3.1 Study Design

A between subject design was used to answer our research questions. The instructor started by informing all participants about the purpose of the study and emphasized the aim of evaluating the material rather then the individual students knowledge. We obtained the consent of every student. For minors we obtained the consent of their parents. The study was run in Germany. For answering our research questions we send an instructor into the partaking school to administer the different training formats to the participants. The instructor had an active part in instructor-based training and was passive in computer-based training as well as text-based training. The course of the user study was split into the following phases:

Pre-Questionnaire: Participants were asked to fill out a pre-questionnaire which contained 16 screenshots of webpages in a randomized order. Eight of these screenshots had been altered to show a phishing URL in the adress bar while the other eight screenshots showed legitimate URLs. Every URL spoofing category was used twice. Participants were asked the following questions for each screenshot: (1) Is the screenshot showing a phishing website or the legitimate website? (2) How certain are you with your decision?

Training: We followed with the specific training format. The duration of text-based training and computer-based training differed between participants. While instructor-based training took exactly 45 min the other two formats differed between participants. When participants had finished they received the Post-Questionnaire.

Post-Questionnaire: The post-questionnaire contained 32 screenshot of webpages (Table 1) in a randomized order with 16 already shown in the pre-questionnaire and 16 new ones. As in the pre-questionnaire we added eight legitimate and eight phishing screenshots again using every URL spoofing category twice. Therefore we ended with 16 phishing, 16 legitimate screenshots and utilized every URL spoofing category four times.

General Survey: Following the post-questionnaire, participants were asked to answer socio-demographic questions. We also included three statements based on the system usability scale (SUS) [3] that were to be answered via a 5 point Likert scale, namely:

- I enjoyed learning about phishing the way I did.
- I think I learned a lot.
- What I learned will help me protecting myself in the future.

3.2 Recruitment

We recruited our participants at a school. School settings offer access to groups of participants that are homogenous in terms of sociodemographic factors like age and educational level. For this purpose we contacted a vocational0 school which has over one thousand students in total split over different professionalisation branches. The school management valued our study as complementary to the schools curriculum and cooperated with us by allowing us to use school hours of 90 min to carry out both the training as well as the evaluation. This had an impact on students motivation, it can be expected to make the results more transferable to education in other contexts where participants are obliged to take part in IT-security trainings (e.g. company context). The user study was carried out in three different classes which were randomly assigned to one of the three training formats. All participants taking part in this study were recruited from the branch of information assistants.

Table 1. Legimitate and phishing URLs used in pre/post-questionnaires.

	Pre and post questionnaire	Post questionnaire only
Original	https://www.chefkoch.de/login.php	https://www.amazon.de/Angebote/b/...
	https://www.ebay.de/rpp/Deals/reisen-...	https://epaper.bild.de/
	https://www.gmx.net/produkte/mail/...	https://secure.ikea.com/webapp/wcs/...
	https://plus.google.com/u/0/me	https://touch.linkedin.com/login.html
	https://www.m.spiegel.de/panorama/...	https://www.stepstone.de/5/index.cfm...
	https://www.blumen.tchibo.de/login...	https://tagesschau.de/frontpage:.
	https://www.t-online.de/wetter/...	https://www.welt.de/sonderthemen/...
	https://blog.xing.com/category/german/	https://de.yahoo.com/...
Phishing	IP/Random URL	
	https://www.lhjwrpik.com/signin/Raum...	https://www.lesen.de/abo/digital
	https://130.83.162.6/signup/	https://198.176.23.15/Ip/pw/login
	Subdomain/Path	
	https://badcat.com/mobile.twitter.com/...	https://events-ma.de/www.gutefrage.net...
	https://web.de.emailclient.com/	https://login.live.dub123.com/login.srf...
	Name Extension	
	https://www.paypal-sicher.com/web...	https://www.zalando-zahlungsarten.de/...
	https://www.shopping-esprit.de/cgi/h2/...	https://de.wikipedia-login.org/index...
	Spelling	
	https://www.maxdorne.de/?fwe=true&...	https://www.0tto.de/damenmode/...
	https://www.windows.mircosoft.com/de...	https://id.sueddeutsche.de/login

4 Results

In total, 81 participants participated. We recruited all participants from the same school and 33 participants had a secondary school leaving certificate, 45 had a high school qualification and three participants had a university degree. The group that took part in instructor-based training had 30, computer-based training 25 and text-based training 26 participants. The instructor-based training group had 2 female and 28 male participants with a mean age of 17.33 (±1.06) and the computer-based training group had 10 female and 14 male participants with a mean age of 20.48 (±4.82). The text-based training group had 2 female and 24 male participants with a mean age of 21.62 (±5.177).

Research question 1 - Effectiveness: Starting with (a) the effectiveness in knowledge transfer of the different formats and (b) the differences in effectiveness between the formats, we measured effectiveness as the number of correct answers for both phishing URLs, legitimate URLs and overall.

(a) First of all we look at the general effectiveness in knowledge transfer. Therefore we evaluated the mean difference between pre-questionnaire and post-questionnaire. A repeated measures ANOVA determined that the mean correct answers differed statistically significant between pre- and post-questionnaire ($F(1, 78) = 3918.92$, $P < 0.001$, $\eta^2 = .98$).

(b) Likewise to the general effectiveness in knowledge transfer the training format (Fig. 4) showed a statistically significant difference for the mean differences ($p < .001$, $\eta^2 = .255$).

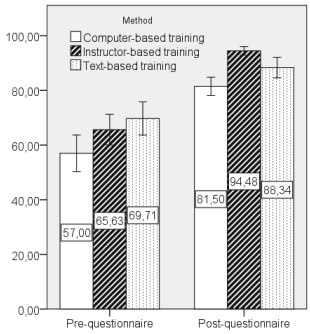


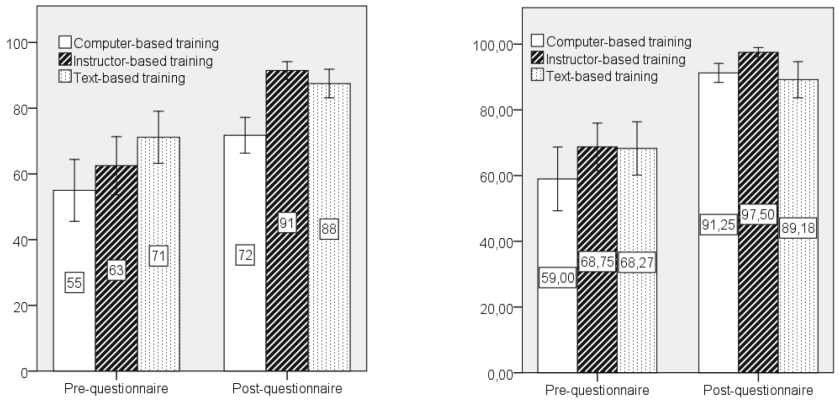
Fig. 4. Mean correct answers for pre & post legimitate and phishing URLs with 95 % confidence interval (in %).

Post-hoc tests for overall correct answers using the Games Howell correction, for different number of participants per group and violated homogeneity of variance, revealed that the difference between computer-based training and instructor-based training is statistically significant ($p < .001$). The same goes for

computer-based training and text-based training ($p = .004$). Whereby instructor-based training and text-based training do not differ significantly from each other ($p = .445$).

Looking at the results of the repeated measures ANOVA, every training showed a significant improvement in detecting both phishing and legitimate URLs as their representative part. Taking this into account post-hoc tests showed further significant differences between the three formats. The instructor-based training achieved both the highest post score for correct answers and the highest improvement in score from pre-questionnaire to post-questionnaire. Nevertheless only looking at the correct answers they do not score significantly better than the text-based training.

Further analyzing the results separated into phishing (Fig. 5a) and legitimate (Fig. 5b) URLs, there is a change. While the results for phishing URLs remain the same, as computer-based training differs statistically from instructor-based ($p < .001$) and text-based training ($p < .001$) and instructor-based training does not differ significantly from text-based training ($p = .997$), this is not the case for legitimate URLs. Only considering these computer-based training remains significant below instructor-based training ($p = .009$). This time there is no statistical significant difference between text-based training and both computer-based ($p = .843$) and instructor-based training ($p = .108$).



(a) Phishing URLs

(b) Legitimate URLs

Fig. 5. Mean correct answers for pre & post split into phishing and legimitate with 95 % confidence interval (in %).

Research question 2 - User satisfaction: We looked at (a) the satisfaction for each format and (b) the differences between the three formats.

(a) As shown in Table 2 all formats achieve high results for overall satisfaction (from 4.09 to 4.46). Split into the questions, starting with the first one,

instructor-based training achieved the highest mean score with ($4.57 \pm .568$), followed by text-based ($4.38 \pm .637$) and computer-based training ($4.2 \pm .764$). For question two instructor-based training again achieved the highest mean score with ($4.34 \pm .814$), this time next is computer-based ($3.88 \pm .971$) and text-based training (3.65 ± 1.198). For the third question the order is instructor-based training with a mean score of ($4.47 \pm .776$), text-based (4.31 ± 1.011) and computer-based training ($4.20 \pm .816$).

Table 2. User satisfaction split into three questions and overall per format.

Format	1. Enjoyed?	2. Learned?	3. Protection?	Overall
Instructor-based training	4.57 (.57)	4.34 (.81)	4.47 (.78)	4.46
Computer-based training	4.2 (.76)	3.88 (.97)	4.2 (.82)	4.09
Text-based training	4.38 (.64)	3.65 (1.2)	4.31 (1.0)	4.11

(b) Starting with the differences for the three training formats over all three questions the one-way ANOVA suggest that there is no statistically significant differences between the training formats ($F(2, 80) = 3.023, p = .51$).

Divided into the three questions, starting with "I enjoyed learning about phishing the way I did." The results of the one-way ANOVA showed no statistically significant difference between the training formats ($F(2, 80) = 2.138, p = .125$).

Next participants had to rate the sentence "I think I learned a lot." The results of the one-way ANOVA showed a statistically significant difference between all formats ($F(2, 79) = 3.433, p = .037$). A Games-Howell post-hoc test showed that participants from the text-based training answered significant lower compared to the computer-based training ($p = .74$) and to those in the instructor-based training ($p = .045$). There was no statistically significant difference between the instructor-based training and the computer-based training ($p = .153$).

Finally participants had to rate the sentence "What I learned will help me protecting myself in the future." The results of the one-way ANOVA showed no statistically significant difference between formats ($F(2, 80) = .658, p = .521$).

Research question 3 - Confidence Level: We looked at (a) the impact the training formats have on people's confidence regarding their own abilities in correctly identifying legitimate and fraudulent websites attacks and (b) differences between the three formats?

(a) Therefore we analyzed the confidence level for all formats in between the pre- and post-questionnaire. A repeated measure ANOVA determined that the mean correct answers differed statistically significant between pre- and post questionnaire ($\mathrm{F}(1, 71) = 150.71$, $P < 0.001$, $\eta^2 = .68$).

(b) Just as the difference between the pre- and post-questionnaire the training formats (Table 3) showed a statistically significant difference for the mean differences ($p = .002$, $\eta^2 = .16$).

Table 3. Average user confidence pre-questionnaire and post-questionnaire per format.

Format	Pre	Post	Difference
Instructor-based training	3.22 (.80)	4.73 (.27)	1.51
Computer-based training	3.58 (.79)	4.58 (.52)	1.00
Text-based training	4.12 (.50)	4.67 (.31)	0.55

Post-hoc tests using the Games Howell correction revealed that the difference between text-based training and instructor-based training is statistically significant ($p < .001$). Whereby computer-based training and instructor-based training do not differ significantly from each other ($p = .76$), as well as computer-based and text-based training ($p = .098$).

Looking at the results of the repeated measures ANOVA, every training showed a significant improvement in confidence. Post-hoc tests showed further significant differences between the three formats. The instructor-based training achieved both the highest post confidence, the highest improvement in confidence and achieved a significant higher confidence than the text-based training.

Research question 4 - Time efficiency: For the fourth and final research question we wanted to know (a) how much the training formats increase the correctly recognition of phishing and legitimate URLs per minute spent with the training and (b) the differences between the three formats.

(a) Table 4 shows the mean time efficiency for the formats. Furthermore, it shows that on average for every minute that the instructor-based group spent with the training, they were able to increase the amount of correct answers given by 0.64 on average. For computer-based training it is on average 0.92 more correct answers per minute and for text-based training on average 1.03 more correct answers

Table 4. Time taken and time efficiency per format. Improvement per minute = percentage of improvement of correct answers divided by mean time.

Format	Mean time (minutes)	Improvement per minute
Instructor-based training	45	0.64
Computer-based training	26.5	0.92
Text-based training	18	1.03

(b) A Kruskal-Wallis H test showed that there was a statistically significant difference in time spent between the different trainings, $\chi^2(2) = 35.01, p < 0.001$.

Median in formats computer-based training and text-based training were 26.5 and 18 min; the distributions in the two formats differed significantly ($U = 55.5, Z = -3.17, p = 0.002, \eta^2 = .26$).

Regarding our time efficiency although the instructor-based training had the biggest improvement (+28.85 %) it is only an improvement of 0.64 correct answers per minute spent. The second best improvement was achieved by the training that took part in computer-based training (+24.5 %) which results in an improvement of 0.92 correct answers per minute spent. Finally the text-based training training achieved lowest general improvement (+18.63 %). Considering the time spent for the training they achieve the highest score with an improvement of 1.03 correct answers per minute spent.

5 Discussion

All training methods improve participants ability in phishing detection and their ability in identifying legitimate webpages significantly. Furthermore, they felt significantly more confident in judging webpages after the training. However, the results of the user study show significant differences between the different training formats. Considering efficiency, user satisfaction and confidence, instructor-based training format achieved the best results. It performed significantly better than the other two formats. At the same time it achieved the lowest time efficiency. This result is in line with the findings in [7]. The authors showed that a social situation and a familiar context like for the pupils in our instructor-based group has a positive learning effect. While the improvement in confidence differs, all three formats achieve a very high score around 4.7 of 5. Not surprisingly, instructor-based training takes more time. While we decided to go for 45 min, in a company instructor-based training requires more time, e.g. as participants need to reach the class room and get back to their offices. Thus, while the security level in the company would increase more than with the other levels, it might not be chosen because of the time (and maybe the costs and the lack of flexibility to decide when to learn and to take breaks).

Interestingly, text-based training performs better than computer-based training. With respect to efficiency, user satisfaction and confidence it achieves the second best results. The results of research question 4 indicate that spending some more minutes with the material is likely to improve the results even further. Thus, if time is a limited resource text-based training is the best training format as it clearly achieves the best results in time-efficiency. Furthermore, participants improved also significantly in making proper decisions as well as in detecting phishing webpages.

A possible explanation for this results is that participants were able to chose their own pace when going through the training. While one can miss important messages when lacking concentration during instructor-based training this cannot happen with text-based training. Participants could go back and read earlier explanations again. Furthermore, it might be easier reading the URL letter by

letter when the PDF is displayed at the screen right in front of them other than displayed by a projector on the wall. What would be interesting but has not been tested is the effect it would have if people do not learn the entire content at once but would start one day and come back at a later time.

The results of the computer-based training group indicate, that this method is not particularly more effective or satisfying than the other two formats; however more expensive, as it needs to be developed. The unexpectedly low improvement of the computer-based training group might possibly be affected by the small screen size of the smartphones. Reading through educational material via a small smartphone screen might not be the best way to partake in security training. While on the other hands it offers a potential for even more interactive training formats. Our results for computer-based training are inline with the results in [19]. However, the authors of [12] got a greater effect than instructor-based training. It needs to be studied further, why their results differ.

Limitations: Participants saw a high amount of phishing examples in a short time frame. Such a high frequency exposure to phishing URLs would likely never happen in a real scenario. While this is the typical problem of phishing studies, this is not a major concern for our study as we compare which format performs better in communicating the content of the NoPhish training concept.

While the text-based training and computer-based training usually have the innate option of pausing at any convenient moment, our study design precluded this possibility. Which effort pausing has to the results needs to be studied in future.

The training was given by a motivated instructor. It is possible that the very positive results of the instructor-based training group can in part be attributed to this fact. But this is a limitation with instructor-based training in general that does not specifically apply to our study design.

Students in the school setting are primed for instructor-based training. This is different from a company setting. Thus, using the instructor-based approach in companies may perform less good than in a school.

Our sample is not representative of the general population, as participants were all students in the professionalisation branch of information assistants. Such they brought an above average general knowledge of IT related problems. However, it shows that also those people lack knowledge in phishing security. Furthermore, earlier evaluations on NoPhish showed significant improvements also for lay people.

6 Related Work

In our study, we used the NoPhish training materials, which are freely available in the three formats in question (instructor-based, computer-based, and text-based). All of the formats provide the same content to the user. In the following, we present similar research comparing different IT-security training formats.

Sheng *et al.* [20] compared the effectiveness of different educational materials to identify phishing websites. Their study included two groups using text-based training and one group using computer-based training. They did not include instructor-based training in their comparison. For their text-based solutions they used "three consumer oriented educational web pages from the first page of Google search results using the search query *phishing*" and the cartoon PhishGuru. The computer-based training used the Anti-Phishing Phil game. The formats used in their study provided different content to the users and not all of them are freely available. The authors did not find any difference in the number of users that fall for phishing between the different formats.

Kumaraguru *et al.* [14] reports on the comparison of three formats for improving the users' skills in terms of detecting phishing attacks. They developed two embedded training designs (computer-based training) and another format consisting of simple email security notices (text-based training). The embedded training designs consisted of regularly sent phishing emails. The results of their study indicate that the embedded training designs (computer-based training) were more effective than simple security notices (text-based training).

Khan *et al.* [13] compared different education material formats based on psychological theories, including instructor-based training, computer-based training (traditional and video games), and text-based training (newsletter articles and posters). They categorize instructor-based training as efficient but unattractive. In their opinion, computer-based training has the advantage of allowing users to learn at their own pace, while being resource-extensive and relatively expensive. For the text-based materials they summarize that these are efficient ways to deliver information, but that it cannot be verified if the information was actually read by the user.

Schilliger and Schmid [19] discuss multiple formats from a theoretical point of view. They see advantages in computer-based training. It can efficiently serve to big groups, due to its support of time and location independent learning. Furthermore, it is much easier to track the learners' success. With regard to text-based training, the authors believe that it is best used to remind people of content they already learned before. Concerning instructor-based training, they argue that it should be as short and as memorable as possible to increase the effectiveness.

Reid [18] developed a software program that supports the delivery of important information. He used a commercial set of knowledge level questions and measured the success of his computer-based technique. The results indicate that frequent repetition of training activities increases knowledge retention.

Canova *et al.* [6] did a small scale lab and retention study of the NoPhish android application. They found significant effects of the application when it comes to transferring the information to the participants both immediately after the study aswell as five months after the participants had taken part in the study.

7 Conclusion and Future Work

We conducted a user study comparing the three different training formats computer-based training, instructor-based training and text-based training with respect to their effectiveness, user satisfaction, confidence and time efficiency. We note that every format lead to a significant improvement of participants IT-security knowledge and confidence in handling the tasks. Also, participants of all groups were satisfied with their respective training format. Instructor-based training created the most promising results regarding three of our four research questions (effectiveness, user satisfaction, confidence). However it performed the worst in our fourth research question (time efficiency). The text-based training format performed slightly worse than instructor-based training. However it has applications in scenarios where time is the most pressing matter as it is the most time efficient formats. Though computer-based training performed worse than the other two formats it still provided for a significant improvement in URL detection. Computer-based training via smartphones enables users to learn whenever they want, wherever they want and for any duration they seem fit and it has benefits when it comes to flexibility in application.

While our study focused on the possible gains of different formats of security training we did not evaluate how the required expenditures differ. A realistic cost assessment including creation and maintenance of the materials remains an area of future work. Also we plan the evaluate the retention of the transmitted knowledge in the future. A comparative analysis of the effect small screensizes have on training effectiveness also remains a topic for further research. Another open issue for future work will be to evaluate whether the rules can be implemented into a usable security tool that automatically addresses incoming E-Mails.

Acknowledgement. This work has been developed within the project 'KMU AWARE' which is funded by the German Federal Ministry for Economic Affairs and Energy under grant no. BMWi-VIA5-090168623-01-1/2015. The authors assume responsibility for the content.

References

1. Abras, C., Maloney-Krichmar, D., Preece, J.: User-centered design. In: Bainbridge, W. (ed.) Encyclopedia of Human-Computer Interaction, vol. 37(4), pp. 445–456. Sage Publications (2004)
2. Bada, M., Sasse, A., Nurse, J.R.C.: Cyber security awareness campaigns: Why do they fail to change behaviour?. In: International Conference on Cyber Security for Sustainable Society, pp. 118–131. Global Cyber Security Centre (2015)
3. Brooke, J.: SUS-A quick and dirty usability scale. In: Usability Evaluation in Industry, vol. 189(194), pp. 4–7. Taylor and Francis (1996)
4. Canova, G., Volkamer, M., Bergmann, C., Borza, R.: NoPhish: An anti-phishing education app. In: Mauw, S., Jensen, C.D. (eds.) STM 2014. LNCS, vol. 8743, pp. 188–192. Springer, Heidelberg (2014)

5. Canova, G., Volkamer, M., Bergmann, C., Borza, R., Reinheimer, B., Stockhardt, S., Tenberg, R.: Learn to spot phishing URLs with the android NoPhish app. In: Bishop, M., Miloslavskaya, N., Theocharidou, M. (eds.) Information Security Education Across the Curriculum. IFIP AICT, vol. 453, pp. 87–100. Springer, Heidelberg (2015)
6. Canova, G., Volkamer, M., Bergmann, C., Reinheimer, B.: NoPhish app evaluation: lab and retention study. In: USEC 2015. Internet Society (2015)
7. Das, S., Kim, H., Dabbish, L.A., Hong, J.I.: The effect of social influence on security sensitivity. In: SOUPS, vol. 14. ACM (2014)
8. Desai, M.S., Richards, T., Eddy, J.P.: A field experiment: instructor-based training vs. computer-based training. J. Instr. Psychol. **27**(4), 239 (2000). George Uhlig Publisher
9. Felt, A.P., Wagner, D.: Phishing on mobile devices. USEC 2011, Internet Society (2011)
10. Garera, S., Provos, N., Chew, M., Rubin, A.D.: A framework for detection and measurement of phishing attacks. In: ACM workshop on Recurring malcode, pp. 1–8. ACM (2007)
11. Greg, A., Rasmussen, R.: Global Phishing Survey: Trends and Domain Name Use in 2H2014 (2015). http://internetidentity.com/wp-content/uploads/2015/05/APWG_Global_Phishing_Report_2H_2014.pdf. Accessed 13 March 2016
12. Harrington, S.S., et al.: A comparison of computer-based and instructor-led training for long-term care staff. J. Contin. Educ. Nurs. **33**(1), 39 (2002)
13. Khan, B., Alghathbar, K.S., Nabi, S.I., Khan, M.K.: Effectiveness of information security awareness methods based on psychological theories. Afr. J. Bus. Manage. **5**(26), 10862–10868 (2011). Academic Journals
14. Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L.F., Hong, J., Nunge, E.: Protecting people from phishing: the design and evaluation of an embedded training email system. In: CHI, pp. 905–914. ACM (2007)
15. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254. ACM (2009)
16. Ng, B.Y., Kankanhalli, A., Xu, Y.C.: Studying users' computer security behavior: A health belief perspective. Decis. Support Syst. **46**(4), 815–825 (2009). Elsevier
17. Ramzan, Z.: Phishing attacks and countermeasures. In: Stavroulakis, P., Stamp, M. (eds.) Handbook of Information and Communication Security, pp. 433–448. Springer, Heidelberg (2010)
18. Reid, D.: Knowledge Retention in Computer-Based Training. University of Calgary, Calgary (2001)
19. Schilliger, B., Schmid, R.: Entwickeln einer Awareness-Kampagne für einen sicheren Umgang mit dem Internet an mittelgrossen Berufs-oder Maturitätsschulen. Ph.D. thesis, Hochschule Luzern, Wirtschaft (2010)
20. Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L.F., Downs, J.: Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In: CHI, pp. 373–382. ACM (2010)
21. Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L.F., Hong, J., Nunge, E.: Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish. In: SOUPS, pp. 88–99. ACM (2007)
22. Thorndike, E.L.: The Fundamentals of Learning. Teachers College Bureau of Publications, New York (1932)

On Gender Specific Perception of Data Sharing in Japan

Markus Tschersich[1], Shinsaku Kiyomoto[2], Sebastian Pape[1(✉)], Toru Nakamura[2], Gökhan Bal[1], Haruo Takasaki[2], and Kai Rannenberg[1]

[1] Chair of Mobile Business and Multilateral Security, Goethe University Frankfurt, Frankfurt, Germany
sebastian.pape@m-chair.de
[2] KDDI R&D Laboratories Inc., Saitama, Japan

Abstract. Privacy and its protection is an important part of the culture in the USA and Europe. Literature in this field lacks empirical data from Japan. Thus, it is difficult– especially for foreign researchers – to understand the situation in Japan. To get a deeper understanding we examined the perception of a topic that is closely related to privacy: the perceived benefits of sharing data and the willingness to share in respect to the benefits for oneself, others and companies. We found a significant impact of the gender to each of the six analysed constructs.

1 Introduction

In the Western world privacy and its protection is an important part of the culture. In the United States and especially in Europe people care about their privacy and therefore also the majority of research about privacy is based on data collection in North-America and Europe [1].

Despite some efforts by Asoh et al. [2] and Takasaki et al. [3], literature in the field of privacy lacks empirical data from Japan. Thus, compared to the USA and Europe, it is difficult for foreign researchers to understand the situation in Japan. In Hofstede's taxonomy for cultural patterns [4, 5] individualism versus collectivism as a cultural norm is identified as one important criterion. Japanese culture — inspired by Confucian and Buddhism philosophy — differs from western culture where individualism is emphasized [6] and therefore, the protection of oneself has a higher importance. As opposed to this, people of the Japanese culture try to identify their role within the groups they are interacting with [7]. Societal harmony is very important to Japanese people and they try to avoid troubling others [8]. This leads them to be more open to share personal information. Additionally, not sharing data is also often understood as isolation from a group. As data sharing often has a major impact on one's privacy, the perception of sharing is very relevant for the behaviour with regard to privacy and its protection. Besides general cultural differences, also the role of men and women differ between the western and the eastern society. This matches another criterion set by Hofstede [4, 5], which distinguishes between cultures that emphasize masculinity

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 150–160, 2016.
DOI: 10.1007/978-3-319-33630-5_11

versus femininity. Based on tenets of Confucianism, taking care on major caregiving responsibilities is the traditional role of women in the Japanese culture [9]. Caused by women's responsibilities in society, the concept of privacy could be more unfamiliar for women than for men. Even though the Japanese society is highly influenced by the western culture, it is interesting to see whether women and men differ in the perceived benefits of sharing data for others and their willingness to e.g. share data.

The paper is structured as follows. Section 2 gives an introduction into privacy research in Japan. Further, Sect. 2 describes the research model and the hypotheses. In Sect. 3 the used methodology is explained. Section 4 summarizes the results of the study, followed by a discussion and conclusions in Sect. 5.

2 Background and Hypotheses

2.1 Related Work

The economics of privacy including empirical studies relating to the consumer's privacy calculus have been evolving [10]. Especially, from late 1990, with the emerging of the Internet, empirical studies have focused on online shopping or online banking. From 2003, online search and 2005 social networks were in the focus. Chellappa and Sin [11] and other research papers around the same years are mostly based on online shopping or personalized services, which are almost the same category of applications. Consumers' benefits are mostly financial ones (discounts, points, etc.). But for the usage of social networks the consumers' benefits are not financial ones but non-financial ones including sharing feelings in a group. A study by Lu et al. [12] demonstrated that social adjustment benefits, i.e. the opportunity of establishing a social identity by integrating into desired social groups, can also have an effect on the intended disclosure behaviour.

The "Act on the Protection of Personal Information" was passed in 2003 and has been enforced in Japan since 2005 [13]. The objectives of this act are to balance the usability of personal information and the protection of individual rights and benefits, and to protect any information that could identify an individual. The act is under revision in terms of expanding coverage of intended data and building an auditing system including privacy commissioners. Recently, for adjusting to the change of international privacy attitudes such as the change in the Privacy Guideline by OECD in 2013 [14], a Consumer Privacy Bill of Rights in US in 2012, and the regulation on the protection of individuals with regards to the processing personal data was passed the European parliament, the law reform proposal on privacy in Japan was published in Jun., 2014. Asoh et al. [2, 15] presented users' privacy concerns in recommendation services using personal data in Japan, and showed that service categories (shopping, navigation and healthcare) did not directly affect to users' privacy concerns. Takasaki et al. [3] analysed privacy concerns for on-line personalization services in Japan. However, in this field

there is a lack of empirical data in Japan. This paper intends to show relevant input for designing and implementing privacy protection policies based on empirical data in Japan.

2.2 Hypotheses

Ahead of a potential revelation of personal information humans perform an economic decision-making process [16]. In the so called privacy calculus, users of information systems weigh the potential costs and benefits of sharing their personal information with specific services, institutions, or persons [1, 17–20]. On the risk side, users take into account the privacy concerns as well as the perceived likelihood and damage of a privacy violation.

On the benefit side, users consider positive effects of the revelation of personal information. The benefits highly depend on the type of service a user could reveal personal information to. Literature describes different examples especially from the field of social network sites [21]. However, literature so far focused on personal benefits from the revelation of personal information. A focus on the western world literature might explain all benefits by focusing just on the user itself. For example, Kobsa [22] determined that gender effects on internet privacy concerns could not be clearly established. From our studies, we confirm the gender effects (especially, women with children are more concerned about privacy protection also they tend to show the strong willingness of personalized service usage).

However, in the Japanese culture privacy is a construct that is tied to egotism [23, 24]. Consequently, Japanese users also reflect the benefits for others and also companies. Earlier studies [2, 3, 15] suggest gender effects in Japan. Especially, women with children are more concerned about privacy protection and also tend to show the strong willingness of personalized service usage. Thus, based on the even higher responsibility of Japanese women in society we hypothesize:

- H1a: Female and male Japanese will differ in their perception that they themselves can benefit when they reveal personal information.
- H1b: Female and male Japanese will differ in their perception that others can benefit when they reveal personal information.
- H1c: Female and male Japanese will differ in their perception that companies can benefit when they reveal personal information.

Result of the Privacy Calculus is the intension to reveal personal information [18]. Thus, if users decide to share personal information, they will be willing to reveal for their personal benefit or the benefit of someone else. Also in the willingness to share we expect an impact of the gender due to women's responsibility in the Japanese culture. Therefore, we hypothesize:

- H2a: Female and Male Japanese will differ in their willingness to share personal information for their personal benefit.
- H2b: Female and Male Japanese will differ in their willingness to share personal information for others' benefit.
- H2c: Female and Male Japanese will differ in their willingness to share personal information for companies' benefit.

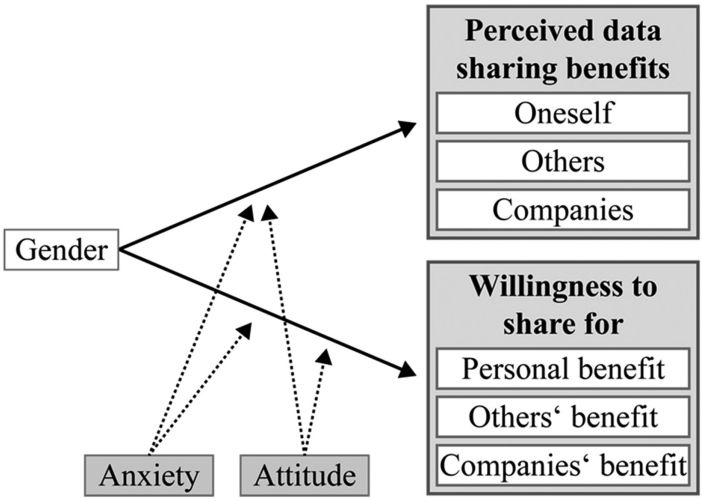


Fig. 1. Overview of the hypotheses and the influencing factors

Users' decisions in the context of computer systems are highly depending on the self-efficacy to specific technologies. Self-efficacy is about what individuals believe about their ability to use a system e.g. effectively [12], and the higher the self-efficacy the greater is the performance achievement. Computer self-efficacy is based on users' attitudes towards computers and their anxiety [25]. Users can differ in their attitude and anxiety and this could bias their decision-making in the privacy calculus and their perceived willingness. Thus, attitude and anxiety are added to our model (cf. Fig. 1) as covariates to address this potential bias.

3 Methodology

3.1 Participants

To get generalizable results, we ran the study with male and female participants with different age and educational background. Overall 9,287 persons participated in the study. For both female and male participants the average age is about 45 years and the distribution to the gender and age groups is also quite equal as shown in Fig. 2.

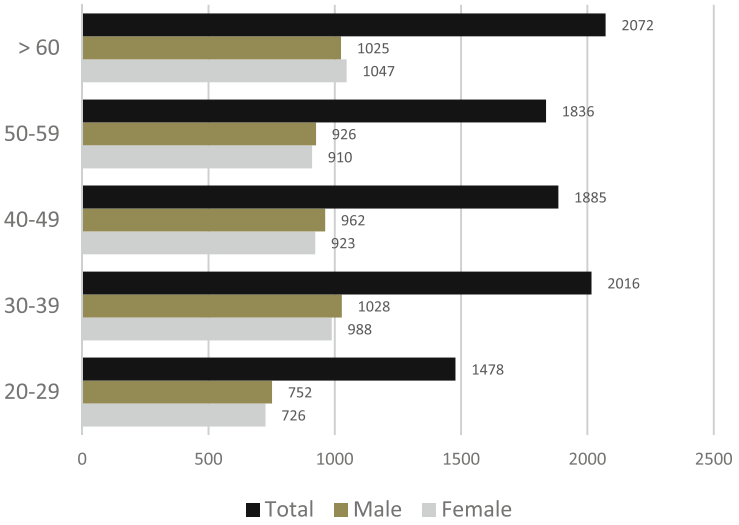


Fig. 2. Gender and age distribution

Participants were collected by publishing an online survey which collected 10000 questionnaires and was executed by a research company. Data was collected from March 12, 2015 until March 19, 2015. We removed responses from people who engaged in research business and advertising business from the participants. We did not select the participants according to where they lived, whether they were married, or whether they had a child, etc.

3.2 Measurement Instrument

An instrument was built to measure impact on the perceived benefits or the willingness to share. The measurement instrument is built out of three to five items for each of the six constructs. As literature and theories do not give items for this construct, we created items by ourselves. Table 1 displays the used items per construct.

The use of new and not tested items increases the need of testing their validity and reliability. Therefore, to test the validity a confirmatory factor analysis (CFA) was performed. The results confirm the items of each construct. Subsequently, the reliability of the scale was tested by checking Cronbach's alpha. According to [20] the alpha values as listed in Tables 2 and 3 are excellent. Further, there was no need to skip items based on the Kaiser-Meyer-Olkin value, because for no construct the dropping of an item increased this value. Thus, we ended up with a measurement instrument including 22 items for the six constructs to be analysed.

3.3 Procedure

The questions were implemented on a web-based questionnaire system by a research company. Selected participants received an instruction for the system from the company, and accessed the system via their own devices. The participants entered their answers via the web-interfaces.

Table 1. Used items per construct

		Female		Male	
		$\overline{x}$	SD	$\overline{x}$	SD
Perceived data sharing benefits	Oneself	3.778	1.002	3.850	1.017
	Others	3.488	1.150	3.677	1.124
	Companies	3.855	1.102	3.999	1.062
Willingness to share	Personal benefit	3.483	1.220	3.751	1.202
	Others' benefit	3.027	1.264	3.361	1.235
	Companies' benefit	2.931	1.266	3.242	1.247

4 Results

4.1 Descriptive Statistics

We analysed the data set and calculated the mean value and the standard deviation (SD) of the descriptive statistics. Both groups, female and male, were handled individually. This allows a comparison of both groups to identify deviations as displayed in Table 1. The comparison shows that the mean values are lower in the case of the female participants than in the case of the male participants. So compared to the male participants the female participants perceived lower sharing benefits in general. Further they have a lower willingness to share personal information.

Further, in the case of the data sharing benefits, all participants perceive most benefits for companies compared to benefits for themselves or even for others. Participants' highest willingness to share personal is for their own benefit. They are less willing to share for others' benefits and even less for the benefit of companies.

4.2 Impact Analysis

To identify the impact of the gender on the different perceived privacy benefits and the willingness to share we run an Analysis of covariance (ANCOVA) for each construct.

Table 2. Constructs on "Perceived privacy benefits"

Construct	Items	Cronbach's alpha
Perceived data sharing benefits ("Me")	Sharing personal information with online service providers can provide me with personalized services tailored to my activity context.	.926
	Sharing personal information with online service providers can provide me with more relevant information tailored to my preferences or personal interests.	
	Sharing personal information with online service providers can provide me with the kind of information or service that I might like.	
	In general, I believe that I can profit from sharing personal information with online service providers.	
	I think that I benefit from sharing personal information with online service providers.	
Perceived data sharing benefits ("Others")	It can provide benefits for other people if I share my personal information with online companies.	.946
	If I share my personal information with online companies, other people can profit from it.	
	In general, I think that it is good for other people if I share my personal information with online companies.	
	It can be useful for other people if I share my personal information with online companies.	
	I'm willing to share personal information with online companies if it is useful for them.	
	I'm in general willing to share personal information with online companies, if I see a benefit for them.	
Perceived data sharing benefits ("Companies")	I believe that online companies can profit from my personal information if I share it with them.	.892
	I believe that it is good for the success of online companies if I share personal information with them.	
	I can support online companies by providing them with my personal information.	
	I think that it is good for online companies if I provide them with my personal information.	

The ANCOVA is used, because this statistical method allows handling the influencing factors "Attitudes" and "Anxiety" (that we had identified in Sect. 2.2) as covariates and so allows filtering out the effect of these influencing factors on the impact. Thus, the bias of attitudes and anxiety of the participants on the result can be prevented, which increases the accuracy of the result. However general linear models like ANCOVA require some assumptions to be fulfilled by the data:

Table 3. Constructs on "Willingness to share"

Construct	Items	Cronbach's alpha
Willingness to share (I benefit)	In general, I'm willing to share personal information with online companies if I can profit from it.	.916
	I'm willing to share personal information with online companies if they provide me with useful services.	
	If I see a benefit for myself, I'm in general willing to share personal information with online companies.	
Willingness to share (others benefit)	In general, I'm willing to share personal information with online companies if other people can profit from it.	.950
	I'm willing to share personal information with online companies if they provide other people with useful services.	
	If I see a benefit for other people, I'm in general willing to share personal information with online companies.	
Willingness to share (companies benefit)	In general, I'm willing to share personal information with online companies if they can profit from it.	.951
	I'm willing to share personal information with online companies if it is useful for them.	
	I'm in general willing to share personal information with online companies, if I see a benefit for them.	

1. The assumption of normality was tested with the Kolmogorov-Smirnov test [26] and reveals that the data of the analysed six constructs is not normal distributed. However, according to the Central Limit Theorem a parametric test can still be used [27].
2. Further, in the case of using ANCOVA the independence of variable (in this case gender) and covariate needs to be checked. The results of the Levene's test [27] indicate that the covariates are independent from each of the analysed six constructs.

The results of the ANCOVA show a highly significant impact of the gender to each of the six analysed constructs. Thus, female and male Japanese people differ in their perception of privacy benefits for oneself, for others and for companies. The same is true for the willingness to share for someone's benefits.

Table 4 shows the result of the ANCOVA. The F-value displays the result of the F-test as a calculation of the quotient of the variance of the answers of the different experimental groups (in our case women and men). This first step allows the next step, which is the check, whether the null hypothesis (H_0), that there would be no significant difference between the different experimental groups (men and women), can be falsified. For this we look at the significance levels (Sig., p-values): A p-value below .05 reveals a significant difference between both experimental groups, a p-value below .001 reveals that the difference is highly significant. Both thresholds are commonly used for

Table 4. Results of the ANCOVA

		Oneself		Others		Companies	
		F	Sig. (p-value)	F	Sig. (p-value)	F	Sig. (p-value)
Perceived benefit for…	Anxiety	135,503	.000	140,548	.000	56,401	.000
	Attitudes	104,633	.000	43,152	.000	79,175	.000
	Gender	27,316	.000	105,988	.000	51,217	.000
Willingness to share for … benefit	Anxiety	114,074	.000	145,547	.000	167,888	.000
	Attitudes	90,705	.000	31,918	.000	29,131	.000
	Gender	148,010	.000	233,217	.000	213,816	.000

this kind of experiments [25]. In our case all the p-values are so small, that they are closer to 0.000 than to 0.001.

The high significance (Sig., p-value < .001) of the covariates "Attitudes" and "Anxiety" as shown in Table 4, demonstrates that the covariates predict the particular dependent variables. Therefore, the perceptions and the willingness are influenced by participants' anxiety and attitudes towards using computer systems. After the effect of anxiety and attitudes has been filtered out by the ANCOVA, the adjusted effect of the gender on the six analysed constructs has a significance value of $p < .001$ as displayed in Table 4 as well. Thus, the statistical tests show that the Hypotheses H1a, H1b, and H1c as well as H2a, H2b, and H2c can be accepted and that the gender of Japanese participants has a significant impact on the perceived benefits and the willingness to share for someone's benefit.

5 Discussion and Conclusion

The results show a highly significant impact of the gender to each of the six analysed constructs. Thus, female and male Japanese people differ in their perception of privacy benefits for oneself, for others and for companies. The same is true for the willingness to share for someone's benefits. Anxiety and attitudes were identified as biases on the measured effects. However gender still has a significant impact on the effects.

Future research should be done to investigate the differences in the perception of privacy in respect to the peoples' culture. It will be interesting if there is a difference between the obtained result for Japan and e.g. Europe or the USA. Additionally, it would be interesting to investigate "Cross-cultural Sharing": Does the willingness to share data with someone belonging to the same culture differ from sharing with someone belonging to another culture?

Further research should also be done regarding the age groups of people. Does the willingness to share data depend on whether they are e.g. Digital Natives or Digital Immigrants [28]?

Finally, practical consequences of our findings should be examined. E.g. would it make sense to offer different services to the different groups?

References

1. Smith, H.J., Dinev, T., Xu, H.: Information privacy research: An interdisciplinary review. MIS Q. **35**(4), 989–1015 (2011)
2. Asoh, H., Ono, C., Habu, Y., Takasaki, H., Takenaka, T., Motomura, Y.: An acceptance model of recommender systems based on a large-scale internet survey. In: Ardissono, L., Kuflik, T. (eds.) UMAP Workshops 2011. LNCS, vol. 7138, pp. 410–414. Springer, Heidelberg (2012)
3. Takasaki, H., Kouguchi, T., Jitsuzumi, T.: A study on causes for privacy concerns about personalized services on mobile devices. J. Public Util. Econ. **2**, 25–34 (2014)
4. Hofstede, G.: Culture's Consequences: comparing values, behaviors, institutions, and organizations across nations, 2nd edn. SAGE Publications, Thousand Oaks, CA (2001)
5. Hofstede, G., Hofstede, G.J.: Cultures and Organizations: Software for the Mind. McGraw Hill Professional, New York (2004)
6. Nakamura, M.: Privacy: Current Status and Pending Issues in Japan. NRI Papers, no. 131 (2008)
7. Miltgen, C.L., Peyrat-Guillard, D.: Cultural and generational influences on privacy concerns: a qualitative study in seven European countries. EJIS **23**(2), 103–125 (2014)
8. Orito, Y., Murata, K.: Privacy protection in Japan: cultural influence on the universal value. In: Electronic Proceedings of Ethicomp (2005)
9. Hashizume, Y.: Gender issues and japanese family-centered caregiving for frail elderly parents or parents-in-law in modern japan: from the sociocultural and historical perspectives. Public Health Nurs. **17**(1), 25–31 (2000)
10. Taylor, C.R.: Consumer privacy and the market for customer information. RAND J. Econ. **35**(4), 631–650 (2004)
11. Chellappa, R.K., Sin, R.G.: Personalization versus privacy: An empirical examination of the online consumer's dilemma. Inf. Technol. Manage. **6**(2), 181–202 (2005)
12. Lu, Y., Tan, B., Hui, K.L.: Inducing customers to disclose personal information to internet businesses with social adjustment benefits. In: Agarwal, R., Kirsch, L.J., DeGross, J.I. (eds.) Proceedings of 25th International Conference on Information Systems, Washington, DC, December 9-12, pp. 272–281 (2004)
13. Japan, Act on the Protection of Personal Information (2003)
14. OECD: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, http://www.oecd.org/internet/ieconomy/oecdguidelinesontheprotectionofprivacyandtransborderflowsofpersonaldata.htm
15. Asoh, H., Takasaki, H., Ono, C., Habu, Y., Takenaka, T., Motomura, Y.: An Analysis on the Intention to Use Recommendation Services Using Lifelogs, no. 4, pp. 846–854 (2015)
16. Awad, N.F., Krishnan, M.: The personalization privacy paradox: An empirical evaluation of information transparency and the willingness to be profiled online for personalization. MIS Q. **30**(1), 13–28 (2006)
17. Culnan, M.: Consumer awareness of name removal procedures: Implications for direct marketing. J. Dir. Mar. **9**(2), 10–19 (1995)
18. Dinev, T., Hart, P.: An extended privacy calculus model for e-commerce transactions. Inf. Syst. Res. **17**(1), 61–80 (2006)
19. Tschersich, M. Botha, R.A.: Understanding the impact of default privacy settings on self-disclosure in social networking services: Building a conceptual model and measurement instrument. In: Proceedings of the 19th Americas Conference on Information Systems, (2013)
20. Kline, P.: The Handbook of Psychological Testing, 2nd edn. Routledge, London (2000)

21. Tschersich, M., Botha, R.A.: Exploring the impact of restrictive default privacy settings on the privacy calculus on social networking sites. In: Presented at the Twenty Second European Conference on Information Systems (2014)
22. Kobsa, A.: Privacy-enhanced personalization. Commun. ACM **50**(8), 24–33 (2007)
23. Ess, C.: Lost in translation intercultural dialogues on privacy and information ethics (introduction to special issue on privacy and data privacy protection in Asia). Ethics Inf. Technol. **7**(1), 1–6 (2005)
24. Kitiyadisai, K.: Privacy rights and protection: foreign values in modern thai context. Ethics Inf. Technol. **7**(1), 17–26 (2005)
25. Torzadeh, G., Doll, W.J.: The development of a tool for measuring the perceived impact of information technology on work. Omega **27**(3), 327–339 (1999)
26. Massey Jr., F.J.: The kolmogorov-smirnov test for goodness of fit. J. Am. Stat. Assoc. **46**(253), 68–78 (1951)
27. Field, A.: Discovering statistics using SPSS, 2nd edn. Sage Publications Limited, London (2006)
28. Prensky, M.: Digital Natives, Digital Immigrants (2001)

TORPEDO: TOoltip-poweRed Phishing Email DetectiOn

Melanie Volkamer[1,3(✉)], Karen Renaud[2], and Benjamin Reinheimer[1]

[1] SECUSO, Computer Science Department, TU Darmstadt, Darmstadt, Germany
{melanie.volkamer,benjamin.reinheimer}@secuso.org
[2] School of Computing Science, University of Glasgow, Glasgow, UK
karen.renaud@glasgow.ac.uk
[3] Karlstad University, Karlstad, Sweden

Abstract. We propose a concept called *TORPEDO* to improve phish detection by providing just-in-time and just-in-place trustworthy tooltips to help people judge links embedded in emails. *TORPEDO*'s tooltips contain the actual URL with the domain highlighted and delay link activation for a short period, giving the person time to inspect the URL before they click. Furthermore, *TORPEDO* consists of an information diagram to explain phish detection. We evaluated *TORPEDO* in particular with respect to its effectiveness: Compared to the worst case 'status bar'. as used in Thunderbird and Web email clients. *TORPEDO* performed significantly better in detecting phishes and identifying legitimate emails (85.17 % versus 43.31 % correct answers for phish). A proof of concept implementation is available as a Thunderbird Add-On.

1 Introduction

Phishing is merely a modern equivalent of a confidence trick that has been carried out for centuries: to deceive someone to derive some personal benefit. The first time that the term "phishing" was used to refer to this digital version of confidence tricking was on January 2, 1996[1]. Phishing messages offer a link embedded in an email message that entices the recipient to click. Email recipients are likely to click on links due to their widespread legitimate use. If they do click, it redirects them to a website masquerading as the real thing or downloads some malware onto their computer. Twenty years after its emergence, phishing still succeeds [11,39]. Automated detection is a powerful tool against phishing, but the fact that it takes, on average, 28.75 h to detect new phish websites [2] means that users have to detect phishing messages themselves during the discovery window. However, many people are unable to distinguish legitimate from phish messages. Since there is no financial bar on the number of emails phishers can send, this means a sizeable number of people are snared every day.

The goal of our research was to propose a solution to reduce phishers' success in the email environment significantly (note, we studied the approach in the

[1] The mention occurred in `alt.online-service.america-online`.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 161–175, 2016.
DOI: 10.1007/978-3-319-33630-5_12

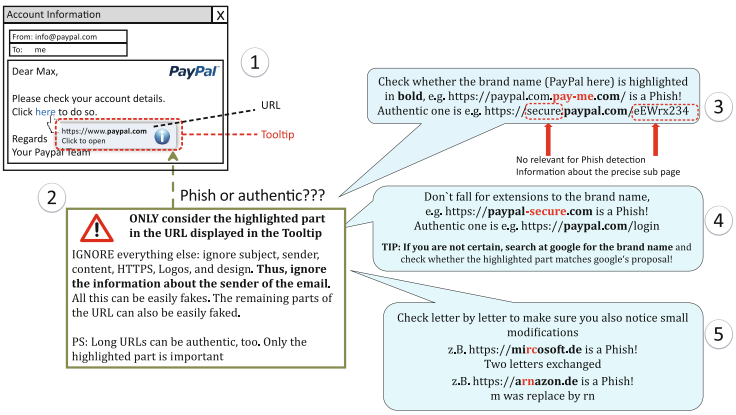


Fig. 1. Just-in-time, just-in-place, trustworthy tooltips as shown in the upper-left part. The entire figure is displayed when more information is requested.

email environment but it could be easily adopted to other message formats such as Facebook and Twitter messages). To achieve this goal, we needed first to understand why people fall for phishing. We thus carried out a literature review and a cognitive walk-through analysis of emails as displayed by commonly-used desktop and webmail clients. Based on our findings, we proposed a concept called *TORPEDO* (**TO**oltip-powe**RE**d **P**hish **E**mail **D**etecti**O**n) to assist users by providing a *just-in-time*, *just-in-place, trustworthy* tooltips that display the actual URL with the domain *highlighted* in bold (see Fig. 1). Furthermore, we disable the link briefly, introducing a short delay, to increase the likelihood that people will check the URL before clicking on it. Finally, we provide users with an *extended information diagram* to explain the phish detection process. An evaluation delivered significant improvements (85.17 % for phish, 91.57 % for legitimate emails compared to 43.31 % and from 63.66 % when only providing the URL in the status bar, as Thunderbird does it for instance). We implemented a corresponding Thunderbird Add-On, as a proof-of-concept.

2 Identifying Ons Why People Fall for Phish

It is important to understand why people fall for phish in order to support them the better. To identify possible reasons for people falling victim to phish, we carried out a literature review and conducted a cognitive walk-through analysis.

2.1 Literature Review

A phishing email contains a number of signals that may indicate that the email is a phish, the most reliable of which is the actual URL as explained in [15,26] Many people do not realise this but, just in case they do, phishers routinely obfuscate the URL to dampen down this signal (e.g. using http://amazon.shop-secure.com

to phish amazon accounts). Our literature review revealed a number of papers in which the authors showed that the reality is different since many people focus on other signals, applying various flawed heuristics, namely:

- *The Sender:* People are likely to trust emails from friends [16] or from reputable businesses [41].
- *The Look and Feel:* People judge emails' trustworthiness based on their first impression, informed by a recognisable logo [4,30], attractive design [30,35], the use of their name or the provision of the company's contact details [17].
- *The Email Content:* People read the email in order to judge the trustworthiness thereof. Relied-upon indicators are the grammar and spelling quality [30,35]. Researchers also showed that people are more likely to fall for a phish when: emotions such as excitement, fear or anxiety [4,35] are provoked, a sense of urgency is invoked [30,34], existing attitudes and beliefs (wanting to believe that the scammer is honest) are exploited [32], or persuasive and influencing techniques are used [35,38]. Researchers argue that under such conditions of arousal people's decision-making abilities are impaired and they are less likely to pick up danger signals [37].
- *Wrong Parts of the URL:* Some people do look at the URL [17]. However, they misinterpret the URL due to a lack of knowledge of the semantics of URLs [9,40]. Some people are reassured by the mere presence of `HTTPS` in the embedded link and look no further [14]. Others are reassured by the brand name being embedded 'somewhere in the URL' [17].

2.2 Cognitive Walk-Through Analysis

For one week, we carefully considered emails that we received, examining the embedded URLs to identify possible challenges which could impair or encourage phish detection. We examined Thunderbird and Apple Mail as well as Web interfaces from three popular Web email clients. We made the following observations:

- *Information not provided where expected/needed:* Thunderbird[2] as well as the Web email clients, display the actual URL destination in the status bar at the bottom of the window. *Problem:* The status bar is some distance away from the user's current attentional focus and might easily be missed. The text of the email is far more prominent and thus likely to be focused on.
- *Tooltip provided by sender:* Some email senders provide a tooltip which appears when the mouse hovers over the link when using Thunderbird or the Web email clients[3] while the actual URL is still displayed in the status bar. Providing such tooltip is actually a very simple attack since the phisher only needs to provide a "title=" attribute.
  Problem: The tooltip encourages examination of the URL by appearing where the user's attention is focused. If the recipient relies only on the tooltip, a phisher would be successful when providing a reassuring URL.

[2] Note, this is different for Apple Mail and also for Outlook.
[3] Again, this is different for Apple Mail and for Outlook.

- *Redirection:* Some email providers seem to make phish detection difficult, if not impossible. These clients do not display the actual URL in the status bar, but rather an (obfuscated) URL, a so-called *dereferer* (see Fig. 2). Web mail providers argue that they do this to protect their users (due to some checks before redirecting users to the actual web page). *Problem:* This approach makes it almost impossible for even the most security aware to detect the perfidy of the link.
- *Tiny URLs:* Some senders of (legitimate) emails use shortened URLs to redirect the person to a different website.
  Problem: From the URL it is impossible to know where a click will send people to. It is necessary to use external services to get the final destination.
- *Habituation:* While Apple Mail shows a toolbar next to the link in the email, it does so (obviously) for both legitimate and phishing emails.
  Problem: While knowing that one should check the URL in the tooltip before clicking, due to habituation people are not very likely to check each URL or even a high percentage.
- *Mouse hover vs. clicking:* In order to get the relevant information in both desktop clients as well as in the Web email clients, one need to touch the link with the mouse while one must not click.
  Problem: It is likely that users are not cautioned enough and instead of only moving the mouse to the link they already click before having checked.

Fig. 2. Status bar displays an obfuscated URL for an embedded URL

2.3 Reasons Why People Fall for Phishing

From the above findings, the following reasons can be deduced:

1. Not being aware that the URL is the only reliable signal: making a decision based on the wrong signal.
2. Not knowing which displayed URL to trust. There are three options: embedded in email text, in the displayed tooltip, or in the status bar.
3. Not having access to the actual URL (destination) due to URLs being obscured – either because of redirection or the use of tiny URLs.
4. Not consulting the URL carefully enough before clicking due to accidental clicks and/or habituation effects.
5. Not knowing how to distinguish authentic from phish URLs.

3 TORPEDO as Possible Solution

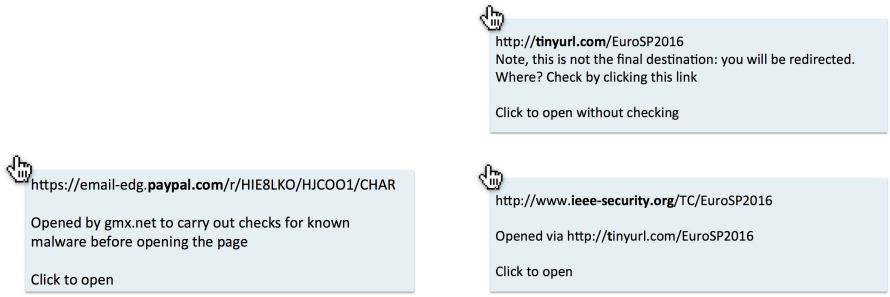
We try to address all of these reasons with *TORPEDO*. *TORPEDO* proves just-in-time and just-in-place trustworthy tooltips which contain the actual URL with the domain highlighted. It delays link activation for a short period. Furthermore, *TORPEDO* consists of an information diagram to explain phish detection, to be used together with the tooltips (when first used and on demand). We explain in the following paragraphs the different aspects and how they are supposed to address the identified reasons.

Just-in-time means that the tooltip appears when the person hovers their mouse over an embedded link. This addresses 'Reason 1' by making the reliable signal more prominent (at least compared to the status bar used by Thunderbird and the Web email clients). *Just-in-place* means that it appears right next to the link (i.e. more precisely right below the link) and only there. This addresses 'Reason 1' and 'Reason 2' by making the most important signal the most prominent one and always displaying it at the same position. *Highlighting the domain* in bold letters (similar to the highlighting in the addressbar of some Web browser) focuses attention on the most important part of the URL addressing 'Reason 5'.

Disabling the link for a short period , perhaps three seconds while providing continuous feedback in terms of a counter showing the time left to click (3, 2, 1s) increases the likelihood of people examining the link before clicking, addressing 'Reason 4'. Note, the delay is configurable to give users control. Furthermore, a white list is maintained to remember domains users have already clicked on twice before (requiring two clicks means that domains will not as easily be accidentally white listed). Whitelisted links will be activated immediately and not be subject to any delay to not annoy users.

Trustworthiness, first, requires overwriting tooltips provided by the email sender. This, together with the tooltip appearing just-in-time and just-in-place, addresses 'Reason 2'. It also addresses 'Reason 3' partially by providing the actual URL, instead of the obfuscated one the phisher wants the user to see. Figure 3(a) shows how we propose to handle the redirections ('redirectUrl=') aspect of 'Reason 3', i.e. providing the actual URL and informing the user that this is the second but final destination. There are two possibilities to address the 'tiny URLs'[4] aspects of 'Reason 3': (1) automatically replace these URLs by the actual one using the service from http://longurl.org, or (2) inform users and let them decide whether to check for the actual URL using this service. From a usability point of view the first option looks more promising; however, from a security and privacy perspective the second one is more promising (as e.g. the tool would send requests although the user does not want to visit the corresponding page). We decided to go for option (2) by default but allowing to configure option (1). Thus, users would first see the upper tooltip of Fig. 3(b) and the other one once decided to check for the actual URL.

[4] According to Wikipedia popular shortening services are: bit.ly, goo.gl, ow.ly, t.co, TinyURL, and Tr.im. The URL is parsed accordingly. Those services are addressed.



(a) Redirection case (b) Shortened URL case

Fig. 3. Example tooltips

Information diagram, to support users in phish detection in general but in particular in checking the URL. This diagram (see Fig. 1) addresses mainly 'Reason 5'. It was iterated several times based on feedback from lay people. The diagram is shown when users initially start using our tooltips. Since users are not regularly confronted with phishing emails they may forget the rules after installation, the diagram is also available on-demand. The information diagram contains the following content while using a process approach explaining step by step what to do while considering the URL manipulation tricks introduced in [7] namely obfuscation, misleading, mangle and camouflage:

- In *Step 1*, we suggest focusing only on the URL. This addresses the fact that people do not focus on the URL ('Reason 1'). It is also explained that the remaining parts of the URL can easily be faked.
- In *Step 2* we recommend that people actually only consider the highlighted part of the URL displayed in the tooltip.
- In *Step 3*, we explain that they should check whether the brand name is highlighted (to address misleading URLs such as http://amazon.shop-secure.com but also obvious phishes such as IP addresses). More precisely, we explain that they should ignore the remaining parts of the URL.
- In *Step 4*, we advise them to check for extensions of the brand name such as in http://amazon-shopping-in-America.com. This the most difficult phishing URL to detect as some companies use such extensions in their authentic domain. We, thus, recommend that they search at Google if they are not sure.
- In *Step 5*, we recommend that they check letter by letter to identify small modifications in the domain name (to pick up http://mircosoft.com).

4 Evaluation

We wanted to evaluate *TORPEDO*'s effectiveness, efficiency and user-engendered confidence in terms of properly judging the authenticity of emails, as compared to the *status quo* status bar display in Thunderbird and the considered Web

email clients. To do so, we conducted a between-subjects online study launched on SoSciSurvey with participants randomly associated to one of two groups:

- **Status bar:** The group sees the URLs in the status bar.
- **TORPEDO:** The group sees the URL in a tooltip with domain highlighted in bold while having seen the information diagram.

Moreover, we formulated the following hypotheses:

- **H1 – Phish detection:** The TORPEDO group will detect more phishing emails, as compared to the status bar group.
- **H2 – Authentic eMail identification:** The TORPEDO group will identify more authentic emails, as compared to the status bar group.
- **H3 – Efficiency:** The TORPEDO group will judge emails more quickly, as compared to the status bar group[5].
- **H4 – Confidence:** The TORPEDO group will be more certain of their judgements, as compared to the status bar group.

4.1 Study Procedure

The study comprised the following three phases[6]:
Phase 1 – Welcome: General information was provided, including the goal of the study, number of phases, the estimated duration, and data protection. We explained that it was important not to seek assistance (we did not elaborate by citing Google, as this could have been counter productive). We introduced the main tasks: They should imagine that their friend Max Müller is about to work through his emails. Since Max has accounts at all the companies in question, it is important for him to know which emails are authentic and which are phish. Therefore, they were asked to help him to judge the emails based on screenshots which Max provides to them on the following pages.
Phase 2 – Judging screenshots: Participants were presented with screenshots of 16 emails (8 authentic / 8 phish) each on a different web service and in random order. The TORPEDO group got in addition the information diagram, both before starting to answer questions as well as below each screenshot. Participants were asked: Is the email authentic? Then participants were then asked: How certain are you that you properly judged the displayed emails. The TORPEDO group was also asked to comment on the information diagram.
Phase 3: Demographics: We requested demographic information.

4.2 Creation of Email Screenshots

We selected 16 web service providers based on the degree of popularity based on Alexa (see Table 1). For all of these, we determined what authentic emails look

[5] Note, on the one hand it is important that people take their time to check the URL, however in addition, if they know what to consider, they can make decisions faster.
[6] Questions were translated from German for this paper.

like (including the 'from' address). All emails addressed the 'Heartbleed-Bug'. The text recommended that the recipient change their password and provided a link to facilitate this. The text slightly differed from one email to the next but the meaning remained the same. All raising some (but not strong) pressure to change the password. Then, half of the screenshots were 'turned into' a screenshot of a phish email by modifying the URL. For the TORPEDO group a tooltip was added to display the relevant link. We decided to simulate a worst-case scenario, i.e. advanced phishing emails which can only reliably be detected by checking the URL. We wanted to investigate the difference between the status bar and tooltip, and not the impact of various different signals. All emails were personalised. We also used `HTTPS` for both phish and non-phish displays because we did not want the absence or presence of `HTTPS` to constitute a cue due to the findings in Sect. 2.1. Next, we considered which URL manipulation techniques to apply to get a representative set of manipulated URLs. Researchers have identified different URL manipulation classifications [7,15,25]. We used the categories from [7] with each type's anticipated success depending on how well users understand URLs and the thoroughness of their URL checking:

- *Obfuscate:* The phish URL is composed of an arbitrary name or IP address. Note, the brand name of the authentic website does not appear.
- *Mislead:* The phish URL embeds the authentic name somewhere (e.g. in the subdomain or the path) in order to allay suspicions.
- *Mangle:* The phish URL includes letter substitutions, different letter ordering, or misspelling e.g. `arnazon` instead of `amazon`.
- *Camouflage:* The domain name of the phish URL contains the brand name together with an extension or a different top level domain.

4.3 Ethics, Recruitment, and Incentives

Our University's ethical requirements with respect to respondent consent and data privacy were met. Participants first read an information page on which they were assured that their data would not be linked to their identity and that the responses would only be used for study purposes. Furthermore, using SoSciSurvey ensured that data was stored in Germany and thus subject to German data protection law. No debriefing was necessary. We recruited participants through a platform called Workhub, which is a German equivalent of Amazon Mechanical Turk. Every Workhub participant receives €3.

5 Results and Discussion

The demographics are summarized for both groups in Table 2. Participants in the TORPEDO group, on average, detected phishing emails 85.17 % of the time and they, on average, identified legitimate emails as such 91.57 % of the time.

Table 1. Legitimate(L) and Manipulated(M) URLs incl. type of manipulation.

Brand	URL (abbreviated with '...')
Postbank	L: https://banking.postbank.de/rai/login
Ebay	L:https://signin.ebay.de/ws/eBayISAPI.dll?SignIn\&UsingSSL...
Xing	L: https://login.xing.com/login?dest_url=https%3A%2F%2Fwww...
Google	L: https://accounts.google.com/login?hl=de
Dropbox	L: https://www.dropbox.com/s/VPrize8EppElIxxWOwETRB87Pe733AR...
Telekom	L: https://accounts.login.idm.telekom.com/oauth2/auth?response...
Zalando	L: https://www.zalando.de/login/
MediaMarkt	L: https://www.mediamarkt.de/webapp/wcs/stores/servlet/Logo...
Facebook	L: https://www.facebook.com/login
(Obfuscate)	M: https://130.83.167.26/login
Flickr	L: https://login.yahoo.com
(Obfuscate)	M: https://www.xplan.com/signing/flickr/
Twitter	L: https://twitter.com/login
(Mislead)	M: https://twitter.webmessenger.com
Amazon	L: https://www.amazon.de/ap/signin
(Mislead)	M: https://www.amazon.de.buecherkaufen.de/ap/signing?...
DeutscheBank	L: https://www.deutsche-bank.de
(Mangle)	M: https://meine.cleutsche-bank.de/trxm/db/init.do?login...
Maxdome	L https://www.maxdome.de/
(Mangle)	M: https://www.maxdorne.de/?fwe=true\&Force-login-layer=true
Paypal	L: https://www.paypal.com/signin/?country.x=DE\&locale...
(Camouflage)	M: https://www.paypalsecure.de/webapps/mpp/home
GMX	L: https://www.gmx.net
(Camouflage)	M: https://meinaccount.gmxfreemail.de/

The corresponding percentages for the control group are: 43.31 % for phish detection and 63.66 % for identifying legitimate emails. Note, participants, on average, detected Camouflaged URLs 72.09 % of the time in the TORPEDO group and 38.37 % in the status bar group, Misleading URLs 87.21 % versus 30.23 %, Mangled URLs 91.86 % versus 37.20 %) and Obfuscated URLs 89.53 % versus 67.44 %. Furthermore, the corresponding numbers for the answer 'I do not know' are (TORPEDO/status bar): 3.49 %/6.98 %, 2.91 %/7.56 %, 2.33 %/8.14 %, and 5.81 %/5.23 %. The descriptive data for H3 and H4 is depicted in Fig. 4.

As to the violation of homogeneity of variances for the compared groups we started our analyses with Mann-Whitney U tests for every hypothesis supplemented with an approximated effect size.

H1 – Phish Detection: Our analysis shows a significantly improved detection rate for phish Emails for participants in the TORPEDO group, as compared to those in the statusbar group ($U = 210, p < .001, \eta^2 = 0.455$).

Table 2. Demographics

	# Participants	Average Age	Median	Youngest	Oldest	Male	IT expert
Status bar	43	25.70	23	17	54	25	5
TORPEDO	43	27.86	26	18	60	25	2

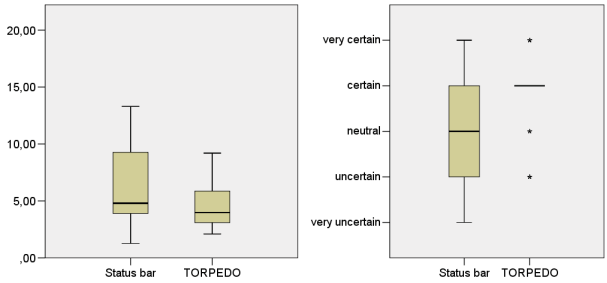


Fig. 4. Descriptive data for timing and certainty of correct decision.

H2 – Authentic Email Identification: Our analysis shows a significantly improved identification rate of authentic Emails for participants in the TORPEDO group, as compared to those in the status bar group ($U = 304, p < .001$, $\eta^2 = 0.374$).

H3 – Efficiency: Our analysis shows that participants in the TORPEDO group were significantly more efficient than participants in the status bar group ($U = 676.5, p = .032$, $\eta^2 = 0.053$).

H4 – Confidence: Our analysis shows that participants in the TORPEDO group were significantly more certain about their decisions than participants in the status bar group ($U = 536.5, p < .001$, $\eta^2 = 0.155$).

Diagram Feedback. We used open coding to analyse the free text answers. We came up with the following categories: 'grateful', 'nothing to improve', 'confusing', 'too much information', 'too little information', and 'small improvements'. Most of the participants (29 from 43 in total) were happy with the diagram, not mentioning anything to improve. Three were grateful. Eight mentioned that the diagram was confusing and five added that the confusion cleared once they read it. Two participants considered the diagram to contain too much information while another two participants complained about it giving too little information (requesting more examples). Three provided small suggestions for improvement: provide a title, and reconsider the usage of terms such as phish and URL.

Discussion. The results show that, in the studied scenario, we significantly improved phish detection as well as the identification of legitimate emails with TORPEDO. The detection rates for all four phishing types increased, too. The participants in the TORPEDO group are also more confident that they made

the proper decision in comparison to the status bar group. The decision making process is also significantly more efficient. Operating more quickly can, in the long run, lead to more errors being made. It is worth providing people with information such as that given in the information diagram since it is easy to apply and requires the email recipient to spend less time checking each individual email. The feedback to the information diagram showed that there is not much need to improve the diagram other than making the numbers clearer and adding a title. We acknowledge that the diagram might not provide sufficient information for some people. In these few cases, we recommend extending our defence approach with existing proven training approaches (see Sect. 6).

Limitations. We acknowledge that we evaluated the approach in a best-case scenario as their primary task was security. Phishing effectiveness evaluations in field studies are not possible due to ethical and legal constraints. Lab studies also have their limitations because participants would not use a study laptop instead of their own. We also acknowledge that the URLs were displayed the entire time and not only when hovering the mouse over the link. This only partially simulates the proposed delay. Note, we only used `HTTPS` since we wanted to assess their ability to check the actual URL, not the presence or absence of `HTTPS`. Finally, we acknowledge that the sample was not representative.

6 Related Work

Researchers have proposed different ways of addressing phishing:

Automated Detection. Phishing emails can be detected either pre- or post-click. Emails can be analysed by the email provider before being forwarded to the user. This analysis includes checking the integrated URLs against several blacklists provided by companies such as Microsoft, Google and phishTank. Other checks can also be carried out. For example, to look at differences between displayed and actual domain names [12] or carry out an NLP analysis of the actual email text [36]. If the email is delivered and the person clicks, post-click detection can also occur. Web browsers or Add-ons can check the URL against various blacklists or check the web site content in combination with the actual URL. A number of different approaches for these checks have been proposed [3,27,28,31]. In both pre- and post-click checks a risky situation can either lead to blocking or a warning e.g. [24,29,40,42]. As a final comment, there is an inherent flaw to post-click warnings. The human tendency to consistency makes it less likely for people to even want to detect the deceptive nature of any site if they have already committed to the process [8]. They have judged the email to be legitimate. Withdrawing at this stage is unlikely. TORPEDO does not aim to replace detection approaches but to complement them in order to help people to protect themselves in case none of the checks detects the phish or it is simply during the pre-detection window [2].

Training. A number of researchers have focused on training users to spot phish [1,5,6,18,20–22,33] but most of them address phish detection in a web

browser context. Researchers have shown that training improves phish detection rates. Training has two drawbacks as compared to TORPEDO. First, people need to be aware that there is a problem and that they need to be active in dealing with it. Otherwise they will not undergo training. This problem was addressed by Kumaraguru [23] by employing the concept of teachable moments, where people are given instructions or training when they almost fall for a phish. In their scheme, instead of blocking a link they allow it, and then show them that they almost fell for a phish. Second, people may forget the information the training imparted as they are not confronted with phishing emails every day. Again the teachable moment approach can help. However, we think providing the information graphic on demand, as and when required, is the safer approach as it might be that the next time they forget how to check the teachable moment mechanism may not be installed and they would be unprotected. Dodge *et al.* [10] report a different approach, post-click training. They send out fake phish messages and then train people who click on the links. They report a positive effect. However, this approach can only be taken within an organisation. A few participants in our evaluation had trouble understanding our diagram. For those, more exhaustive training may help them. Note, the training, as such, would be shorter than what would be required without TORPEDO.

Combining Approaches. We propose TORPEDO to complement existing approaches to address the email phishing problem. Other researchers have also proposed combining approaches to maximise phish protection. Khonji *et al.* [19] suggest a two-pronged approach, the first prong being user training, and the second being automated detection. The latter includes blacklists, machine learning and visual similarity detection. Frauenstein and Von Solms [13] propose combining human, organisational and technical measures. The first includes awareness and training, the second policies and procedures and the last one includes automated measures to detect phish.

7 Conclusion and Future Work

Phishing is a thorny issue. Trying to filter out phishing emails before they reached the end users reduces the problem. Great strides have been made in this direction but no one will claim that any automated system will catch 100 % of phishing emails. So, it is up to the end users to protect themselves. The research we have presented here offers a way to support end users by deploying TORPEDO providing just-in-time, just-in-place, trustworthy tooltips; disabling links for a short period of time; detection of re-directions and tiny URLs, and providing a diagram at installation, or on demand, that encapsulates phish detection advice in a step by step fashion. This approach was evaluated and improved. We found that it highly significantly improved phish and legitimate email detection, made such detection significantly faster and led to people feeling more confident about their judgements compared to the status bar approach as used in Thunderbird and Web email clients. With 85.17 % phish detection compared to 43.31 % with the status bar URL display, it can only be hoped that the different email clients

and email providers deploy this approach as soon as possible. Until then, people can use the *TORPEDO* Add-on we developed. As future work, we plan to conduct acceptance tests to determine whether TORPEDO will indeed be used. We also plan to extend this approach to mobile email clients.

Acknowledgement. This work was developed within the project 'KMU AWARE' which is funded by the German Federal Ministry for Economic Affairs and Energy under grant no. BMWi-VIA5-090168623-01-1/2015. The authors assume responsibility for the content.

References

1. Alnajim, A., Munro, M.: An anti-phishing approach that uses training intervention for phishing websites detection. In: 6th International Conference on Information Technology: New Generations, pp. 405–410. IEEE (2009)
2. APWG Internet Policy Committee: Global Phishing Survey: Trends and Domain Name Use in 2H2013 (2013). http://docs.apwg.org/reports/APWG_GlobalPhishingSurvey_2H2013.pdf. Accessed 13 March 2016
3. Bar-Yossef, Z., Keidar, I., Schonfeld, U.: Do not crawl in the DUST: different URLs with similar text. TWEB **3**(1), 1–31 (2009). ACM
4. Blythe, M., Petrie, H., Clark, J.A.: F for fake: four studies on how we fall for phish. In: CHI, pp. 3469–3478. ACM (2011)
5. Canova, G., Volkamer, M., Bergmann, C., Borza, R.: NoPhish: an anti-phishing education app. In: Mauw, S., Jensen, C.D. (eds.) STM 2014. LNCS, vol. 8743, pp. 188–192. Springer, Heidelberg (2014)
6. Canova, G., Volkamer, M., Bergmann, C., Borza, R.: Learn to spot phishing URLs with the android nophish app. In: Bishop, M., Miloslavskaya, N., Theocharidou, M. (eds.) Information Security Education Across the Curriculum. IFIP Advances in Information and Communication Technology, vol. 453, pp. 87–100. Springer, Heidelberg (2015)
7. Canova, G., Volkamer, M., Bergmann, C., Reinheimer, B.: NoPhish app evaluation: lab and retention study. In: USEC. Internet Society (2015)
8. Cialdini, R.B., Cacioppo, J.T., Bassett, R., Miller, J.A.: Low-ball procedure for producing compliance: commitment then cost. J. Pers. Soc. Psychol. **36**(5), 463 (1978). APA
9. Dhamija, R., Tygar, J.D., Hearst, M.: Why phishing works. In: CHI, pp. 581–590. ACM (2006)
10. Dodge, R.C., Carver, C., Ferguson, A.J.: Phishing for user security awareness. Comput. Secur. **26**(1), 73–80 (2007). Elsevier
11. Erkkilä, J.-P.: Why we fall for phishing. In: Conference on Human Factors in Computer Systems. ACM (2011)
12. Fette, I., Sadeh, N., Tomasic, A.: Learning to detect phishing emails. In: 16th International Conference on World Wide Web, pp. 649–656. ACM (2007)
13. Frauenstein, E.D., von Solms, R.: Phishing: how an organization can protect itself. In: Information Security South Africa Conference, pp. 253–268. Information Security South Africa (2009)
14. Friedman, B., Hurley, D., Howe, D.C., Felten, E., Nissenbaum, H.: Users' conceptions of web security: a comparative study. In: CHI, pp. 746–747. ACM (2002)

15. Garera, S., Provos, N., Chew, M., Rubin, A.D.: A framework for detection and measurement of phishing attacks. In: Recurring Malcode, pp. 1–8. ACM (2007)
16. Jagatic, T.N., Johnson, N.A., Jakobsson, M., Menczer, F.: Social phishing. Commun. ACM **50**(10), 94–100 (2007). ACM
17. Jakobsson, M., Tsow, A., Shah, A., Blevis, E., Lim, Y.: What instills trust? a qualitative study of phishing. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 356–361. Springer, Heidelberg (2007)
18. Jansson, K., von Solms, R.: Simulating malicious emails to educate end users on-demand. In: 3rd Symposium on Web Society, pp. 74–80. IEEE (2011)
19. Khonji, M., Iraqi, Y., Jones, A.: Phishing detection: a literature survey. Commun. Surv. Tutorials IEEE **15**(4), 2091–2121 (2013). IEEE
20. Kirlappos, I., Sasse, M.A., Education, S.: Against phishing: a modest proposal for a major rethink. Secur. Priv. **10**(2), 24–32 (2012). IEEE
21. Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L.F., Hong, J., Nunge, E.: Protecting people from phishing: the design and evaluation of an embedded training email system. In: CHI, pp. 905–914. ACM (2007)
22. Kumaraguru, P., Rhee, Y., Sheng, S., Hasan, S., Acquisti, A., Cranor, L.-F., Hong, J.: Getting users to pay attention to anti-phishing education: evaluation of retention and transfer. In: Anti-phishing WG, pp. 70–81. ACM (2007)
23. Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L.F., Hong, J.: Teaching Johnny to fall for phish. Trans. Internet Technol. **10**(2), 1–7 (2010). ACM
24. Li, L., Helenius, M.: Usability evaluation of anti-phishing toolbars. J. Comput. Virol. **3**(2), 163–184 (2007). Springer
25. Lin, E., Greenberg, S., Trotter, E., Ma, D., Aycock, J.: Does domain highlighting help people identify phishing sites? In: CHI, pp. 2075–2084. ACM (2011)
26. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists learning to detect malicious web sites from suspicious URLs. In: 15th SIGKDD, pp. 1245–1254. ACM (2009)
27. Marchal, S., François, J., State, R., Engel, T.: Proactive discovery of phishing related domain names. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 190–209. Springer, Heidelberg (2012)
28. Maurer, M.-E., Herzner, D.: Using visual website similarity for phishing detection and reporting. In: CHI, pp. 1625–1630. ACM (2012)
29. Maurer, M.-E., Luca, A.D., Kempe, S.: Using data type based security alert dialogs to raise online security awareness. In: SOUPS, p. 2. ACM (2011)
30. Naidoo, R.: Analysing urgency and trust cues exploited in phishing scam designs. In: 10th International Conference on Cyber Warfare and Security, p. 216. Academic Conferences Limited (2015)
31. Prakash, P., Kumar, M., Kompella, R.R., Gupta, M.: PhishNet: predictive blacklisting to detect phishing attacks. In: INFOCOM, pp. 1–5. IEEE (2010)
32. Rusch, J.J.: The "social engineering" of internet fraud. In: Internet Society Annual Conference. Internet Society (1999)
33. Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L.F., Hong, J., Nunge, E., Phil, A.-P.: The design and evaluation of a game that teaches people not to fall for phish. In: SOUPS, pp. 88–99. ACM (2007)
34. Stajano, F., Wilson, P.: Understanding scam victims: seven principles for systems security. Commun. ACM **54**(3), 70–75 (2011). ACM
35. University of Exeter School of Psychology. The psychology of scams: Provoking and committing errors of judgement, University of Exeter (2012)

36. Verma, R., Shashidhar, N., Hossain, N.: Detecting phishing emails the natural language way. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 824–841. Springer, Heidelberg (2012)
37. Vishwanath, A., Herath, T., Chen, R., Wang, J., Rao, H.R.: Why do people get phished? testing individual differences in phishing vulnerability within an integrated, information processing model. Decis. Supp. Syst. **51**(3), 576–586 (2011). Elsevier
38. Wang, J., Chen, R., Herath, T., Rao, H.: An empirical exploration of the design pattern of phishing attacks. Inform. Assurance, Security & Privacy Services, Emerald Publishers (2009)
39. Webroot. Webroot 2015 Threat Brief. http://www.webroot.com/shared/pdf/Webroot_2015_Threat_Brief.pdf. Accessed 13 March 2016
40. Wu, M., Miller, R.C., Garfinkel, S.L.: Do security toolbars actually prevent phishing attacks? In: CHI, pp. 601–610. ACM (2006)
41. Xu, Z., Zhang, W.: Victimized by phishing: a heuristic-systematic perspective. J. Internet Bank. Commer. **17**(3), 1 (2012). ARRAY Development
42. Zhang, Y., Egelman, S., Cranor, L.F., Hong, J.: Phinding phish: evaluating anti-phishing tools. In: NDSS. School of Computer Science, Internet Society (2007)

Social Networks

SybilRadar: A Graph-Structure Based Framework for Sybil Detection in On-line Social Networks

Dieudonné Mulamba(✉), Indrajit Ray, and Indrakshi Ray

Department of Computer Science, Colorado State University,
Fort Collins, CO 80523, USA
{mulamba,indrajit,iray}@cs.colostate.edu

Abstract. Online Social Networks (OSN) are increasingly becoming victims of Sybil attacks. These attacks involve creation of multiple colluding fake accounts (called Sybils) with the goal of compromising the trust underpinnings of the OSN, in turn, leading to security and the privacy violations. Existing mechanisms to detect Sybils are based either on analyzing user attributes and activities, which are often incomplete or inaccurate or raise privacy concerns, or on analyzing the topological structures of the OSN. Two major assumptions that the latter category of works make, namely, that the OSN can be partitioned into a Sybil and a non-Sybil region and that the so-called "attack edges" between Sybil nodes and non-Sybil nodes are only a handful, often do not hold in real life scenarios. Consequently, when attackers engineer Sybils to behave like real user accounts, these mechanisms perform poorly. In this work, we propose SybilRadar, a robust Sybil detection framework based on graph-based structural properties of an OSN that does not rely on the traditional non-realistic assumptions that similar structure-based frameworks make. We run SybilRadar on both synthetic as well as real-world OSN data. Our results demonstrate that SybilRadar has very high detection rate even when the network is not fast mixing and the so-called "attack edges" between Sybils and non-Sybils are in the tens of thousands.

Keywords: Security of on-line social networks · Sybil attacks and detection · Graph structures · Graph metrics

1 Introduction

The success of Online Social Networks (OSNs) [11] such as Facebook, Twitter, LinkedIN, and Google+, have made them a lucrative target for attackers. Owing to their open nature, they are specifically vulnerable to a new form of threat call Sybils. In a Sybil attack, an adversary creates a large number of fake identities or forges a large number of existing identities and uses those to target the trust underpinnings of the OSN [7]. Various types of malicious attacks can be launched

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 179–193, 2016.
DOI: 10.1007/978-3-319-33630-5_13

this way such as, social spamming [30], malware distribution [33], and private data collection [5]. Therefore, it is important to provide OSN administrators a tool for detecting Sybil accounts automatically, speedily and accurately.

Although much effort has been devoted to design such a tool, existing Sybil defense approaches are efficient against a naïve attacker but can be evaded by sophisticated ones. An attacker can evade a "content-based approach" in which different features of OSN user-level attributes and activities are analyzed to discriminate them from fake account activities [28], by creating fake accounts whose features are similar to those of real accounts. On the other hand, several researches [10,17,34] have shown that "structure-based approaches", which model the OSN as graph with nodes and edges respectively representing user accounts and social relationships, can be evaded by an attacker who succeeds in creating a large number of edges between the fake accounts and the benign ones. This happens specially in weak-trust OSNs. Given that extracting and selecting appropriate features from users attributes and activities for content-based Sybil detection is challenging, prone to inaccuracies, and often raises privacy issues, we propose SybilRadar, a Sybil detection mechanism that is based on graph-based structural properties of OSNs. SybilRadar is able to protect OSNs with weak trust relationships against Sybil attacks. We exprimentally evaluate the accuracy of SybilRadar in detecting Sybils using real world OSN data. Our results show that SybilRadar has much better detection accuracy than the closest competitor.

The rest of the paper is organized as follows. Section 2 discusses major works in OSN Sybil defense. In Sect. 3 we present the system model for SybilRadar. We discuss why assumptions in existing structure-based detection mechanisms are invalid under real world settings. We end the section with a discussion on our attack model. The main design of SybilRadar is presented in Sect. 4. We discuss the major intuitions in our design and the different graph metrics that we used. Section 5 presents the experimental setup and evaluation of SybilRadar including comparison with SybilRank, which is the closest in design to SybilRadar. We conclude in Sect. 6 with a discussion of our results and pointers to future work.

2 Related Works

Several studies have shown that OSNs are very vulnerable to Sybil attacks. Facebook [14], Twitter [20,30], and Renren [34] have each experienced significant amount of spams whose origins were Sybil attacks. Several researchers have investigated approaches to defend against Sybil attacks on Online Social Network following studies that have been conducted to assess the severity of these attacks. Two bodies of works have been proposed in order to mitigate Sybils. The first body of works that we call content-based approaches leverages user behaviors and employs machine-learning techniques to learn and classify these behaviors. OSN nodes deviating significantly from these nodes are called Sybils. The second body of works that we call structure-based approaches leverages graph-theoretic proprieties of the social network. Nodes that exhibit significantly different properties than others are identified as Sybils.

Content-based approaches aim to find Sybil accounts by using a classifier trained using machine-learning techniques. The most recent user activities are analyzed to extract some unique features that will serve as inputs on which a classifier is built. Machine-learning techniques such as clustering, support vector machines, and Bayesian networks are used to build the classifier. Some of these approaches are used for spam detection such as blacklisting, whitelisting, and URL filtering [29,30,32]. While many of these approaches have very high detection rates, the problem with these approaches is that they are only as good as the data that are used to train the classifiers. We believe that identifying proper features from user attributes and activities is challenging because these attributes often contain incomplete, inaccurate and sometimes purposefully misleading information. Additionally, a sophisticated attacker can create fake accounts presenting features similar to the ones one of real accounts, thus evading detection. We also believe creating such user profiles can lead to privacy breaches and are not supportive of such techniques. Consequently, We do not consider content-based approaches in our work any further.

Structure-based approaches model an OSN as a graph with user accounts and social relationships respectively represented by nodes and links. These approaches determine some graph-theoretic characteristics of nodes which are then used to discriminate Sybils from the real ones. Existing structure-approaches are based on two assumptions. The first is that the social graph will be partitioned into two distinct regions, one region with the Sybil nodes and the other one with benign nodes. The second assumption is that there will be only a small number of attack edges between the two regions, as a consequence of the strong trust relationship in the social graph. Several mechanisms use these approaches to detect Sybil communities, which are tight-knit communities that have a small quotient-cut from the honest region of the graph [9,37,38].

SybilRank [6] is one of the most well-known techniques. It uses graph-theoretic properties of the OSN social graph to compute the likelihood of users to be Sybils in order to perform the ranking. The detection starts with the administrator determining some known real users as initial seed node. A short random walk is run with the known seeds. At the end of the random walk, all nodes are given trust values which are the landing probabilities for the random walk. SybilRank then ranks all the nodes based on their trust value. Nodes having higher trust value will be at the top, while the nodes with lower trust values will be lowly ranked. SybilRank performs almost linearly in the size of the social graph.

However, SybilRank is based on certain assumptions that several researches [24,27] have proven not to be true in real life. In addition to these researches, Yang et al. show that Sybils on Renren blend into the social graph rather than forming tight communities [34]. Mohaisen et al. show that many social graphs are not fast-mixing, which is a necessary precondition for the structure-based Sybil detector of SybilRank to be effective [24]. SybilRadar, on the other hand, does not make any of these assumptions.

Integro [4] is an approach that extends SybilRank. It is developed without the two assumptions on which SybilRank is based on. Integro is a hybrid approach.

It mixes content-based approach with a structure-based approach in order to detect Sybils. Integro first determines unique features for users which are used to build a feature-vector. The feature-vectors are used to train a classifier that predicts potential victims of Sybil attacks. After finding the potential victims, the edges in the social graph are given weights based on whether they are adjacent to the potential victims or not. The ranking is then performed by a modified random walk. Integro achieved a 95 % precision in detecting Sybils. Our approach produces similar detection accuracy without using any content-based techniques.

SybilFrame [15] relaxes the assumptions that the social network can be partitioned into two distinct regions – Sybil and non-Sybil – and that there exists only a small number of attack edges between the two regions. SybilFrame is also a hybrid approach that leverages the attributes of an individual node along with a measure of correlation between connected nodes in order to classify nodes among benign and Sybils. SybilFrame operates in two steps. In the first step the initial network data are fed into the framework from which node unique features are extracted in order to compute node prior information. In Step 2, the node prior information are provided to the posterior inference layer in order to compute the correlation between nodes. This nodes correlation is computed using Markov Random Field, and along with the Loopy Belief Propagation method, it provides the posterior information of nodes which is used to perform the ranking of nodes.

3 Preliminaries

We begin by presenting the system model for our work. We then introduce the notion of strong and weak trust relationships in OSNs. We explain why SybilRank does not perform well in a real-world OSN with weak trust. We end this section with a discussion of our attack model.

System Model: Trust relationship between two OSN users allows one to assess the information based upon which further information sharing can be performed or a service can be expected [18], and is the underpinning on which OSNs are built. Consider the social network topology as defined by a graph G = (V,E) comprising a set of vertices V, denoting users on the social network and E a set of edges, representing trust relationships (or friendship) between users. We assume trust relationships are mutual (bi-directional) and represent it with undirected edges between the users in the graph G. Two kind of nodes are considered here – an *honest* node and a Sybil node. A honest node that has accepted, or is susceptible to accepting a friend request from a Sybil node is considered to be a *victim* node. The subgraph of G containing all the honest nodes is considered to be the honest region of the OSN, while the Sybil region is the subgraph of G containing all sybil nodes.

We consider three kind of edges. *Attack* edges are those connecting victim nodes in an honest region and Sybil niodes. *Sybil* edges connet Sybil nodes to each other. Finally we have *honest* edges that connect honest nodes with each other.

OSNs with weak trust: In early studies [25,37], OSNs were assumed to have strong trust relationships. OSNs with strong trust are those that possess the property of *fast-mixing*. For Sybil detection purposes, this boils down to a social network with a *small cut*, which is a set of edges whose removal will disconnect the graph into two distinct regions – the honest region and the Sybil region [39]. In other words, in a social network with strong trust we can distinguish the two distinct regions and there is a very limited number of attack edges between the regions (in the tens). OSN with weak trust, on the other hand, is a network that does not display the fast-mixing property. Indeed, it was demonstrated [24] that not many social networks are fast-mixing. In this work, we assume an OSN with weak trust, which is in contrast to SybilRank.

Attack Model: We assume that an attacker can create an unlimited number of Sybil nodes constituting a subgraph (the Sybil region) whose topology is beyond the control of the OSN provider. Attackers can create as many number of attack edges as they want, but they do not have control on how many of those attacks edges will be *successful* in establishing victims. Our Sybil defense mechanism is built around the assumption that we know at least one honest node. This assumption is reasonable since such information can be provided by the administrator of the OSN after a carefully designed process for that purpose. Same assumption is made by other works as well. In addition, we assume that the attacker does not have complete knowledge of the entire OSN topology, since this will require him to crawl the entire network. However, the attacker can acquire the knowledge about a subgraph of the OSN.

4 SybilRadar System Design

SybilRadar operates in three steps. The process starts with the network dataset (set of nodes and edges) being fed to the SybilRadar framework. The first step involves the computation of similarity values between a given pair of nodes. The chosen similarity metric is the Adamic-Adar metric [1], which is based on the notion of common friends between any given pair of nodes. The intuition for choosing this metric is that honest nodes will have more friends in common that Sybil nodes. In the second step, the result from the first step is refined using another similarity metric which is the Within-Inter-Community metric (WIC) [31]. This metric leverages the underlying community structure of the given social graph. The Louvain method [3] is used to find the social graph community information that is fed to the WIC similarity metric computation. This step produces the prior information which is the similarity values of any given pair of nodes driven by the community they belong to. We end this step with a tuning of the nodes similarity values for those nodes with a similarity value greater than 1. We assign the resulting similarity values to the social graph edges as their weights. In the third step, we run a Modified Short Random Walk on the weighted social graph. This step produces trust values, which are the node's landing probabilities of the random walk. These values are assigned to each node as the posterior information in order to perform the ranking of nodes.

4.1 Predicting Attack Edges

Similarity metrics have been extensively used in the field of link prediction in networks. The link prediction problem consists of predicting possible future links based on observing existing links in a given network. Sybils try to maliciously create trust relationships with honest nodes by creating attack edges. Our algorithm tries to predict these bad links. The prediction of future possible links can be based on observing unique and recent features of nodes present in the network, or can be based on structural properties of nodes present in the network. In the first case, feature similarity metrics are used, while structural similarity metrics are used in the latter case. Interested readers are referred to [21,23] for link prediction works using feature similarity metrics, and references [13,35] for link prediction works based on structural similarity metrics. In OSNs node attributes are not always available. For example, users may not complete their profiles or provide inaccurate or misleading information to protect their sensitive information. Moreover, trying to learn user behavior, where complete information is available, may raise privacy concerns. This leads us to consider structural similarity metrics, which are based solely on the structure of the social graph induced by trust relationships between users [16].

We adopt the Adamic-Adar metric [1] to compute an initial similarity value of pairs of nodes. For a given OSN graph $G = (V, E)$, let x and y be two nodes and $\Gamma(x)$ and $\Gamma(y)$ be the sets of neighbors of x and y. The Adamic-Adar (or simply Adamic) similarity measure is given by

$$S_{x,y}^{AA} = \sum_{w \in \Gamma(x) \cap \Gamma(y)} \frac{1}{log \mid \Gamma(w) \mid} \tag{1}$$

Given the initial social graph, running the *Adamic* similarity metric on each pair of nodes results in a weighted social graph with the weight on a link being the similarity value of nodes adjacent to that link. For a given social graph $G = (V, E)$ and for each edge $(u_1, u_2) \in E$, the similarity value $Adamic(u_1, u_2)$ becomes its weight $w(u_1, u_2)$. After computing the Adamic similarity metric we make the following observations:

1. We have three sets of edges: edges with weight $w(u_1, u_2) = 0$, those with weight $w(u_1, u_2) \in [0, 1]$, and the edges with weight $w(u_1, u_2) > 1$.
2. For the attack edges, at least 95 % of them have their weight $w(u_1, u_2) = 0$, and less than 5 % have their weight $w(u_1, u_2) \in [0, 1]$, while about zero to an infinitely small number of them have their weight $w(u_1, u_2) > 1$.
3. The situation for honest edges is quite different. At least 90 % of them have their weight $w(u_1, u_2) > 1$, and about less than 5 % have their weight $w(u_1, u_2) \in [0, 1]$, whereas those with weights $w(u_1, u_2) = 0$ are also less than 5 %.

We were able to make these observation because the social graph used for simulation purpose is derived from a synthetic network whose attack edges, Sybil edges and honest edges are known beforehand. We were able to predict about

90 % of existing attack edges. We made similar observation later with our real data. We observe that predicting attack edges can be very helpful. since it can reveal nodes that have potentially been victims of Sybil attacks. This can be a valuable information for a system administrator. Note, however, that not all edges that have their $w(u_1, u_2) = 0$ are all attack edges. In other words, some honest edges, as well as some Sybil edges, have their weight equal to 0. This is due to the fact that not all pairs of honest nodes or Sybil nodes have common friends, which is the criteria used in computing the similarity value using the Adamic metric.

4.2 Further Refinement of Attack Edge Detection

We next observed that there was an extreme case where our current Sybil detection algorithm completely looses its accuracy. This situation arises when the number of attack edges far exceeds the number of honest nodes.

This situation is not desirable because, at this level, any attacker that can succeed to create a huge number of attack edges compared to the number of benign accounts and get a high degree of certainty of having a significant number of his Sybil accounts evading the Sybil detection mechanism. We observe that among the attack edges that were not detected a significant number have their weights $w(u_1, u_2) \in [0, 1]$. These edges are mixed with a significant portion of other non attack edges which also have their weight $w(u_1, u_2) \in [0, 1]$. We want to filter out as many attack edges as we can in order to increase the number of detected attack edges. For this purpose, we leverage properties of communities (or clusters) in networks.

Community Detection. OSNs typically display clustering characteristics. The idea of using a clustering structure when designing a similarity metric was advanced by [12] who showed that link prediction measures based on structural similarity perform poorly for a network with a low clustering structure. This inspired [31] to first divide the network into communities, and use this clustering structure information in designing a similarity metric for the link prediction problem. In order to measure the quality of a community structure, Newman et al. [26] introduced a modularity function Q. Given a social graph $G = (V, E)$, the modularity function can be expressed as follows:

$$Q = \frac{1}{2m} \sum (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j) \tag{2}$$

where k_i and k_j are respectively the degree of nodes i and j. A_{ij} represents an element of the adjacency matrix, and m is the size of E which is the set of edges of the given graph G. C_i and C_j are the respective communities to which i and j belong. The parameter δ is the Kronecker delta symbol whose value is 1 when both i and j belong to the same community, and is 0 when both nodes belong to different communities. The goal of community detection is to divide a network into communities in a manner that maximizes the value of the modularity.

In our Sybil detection algorithm we use a modularity optimization method called the Louvain Method [3].

To identify clusters, we first collect all the edges with weight $w(u_1, u_2) \in [0, 1]$, and for each of these edges we compute the similarity value of its end nodes using the Within Inter Cluster (WIC) similarity metric [31]. This metric is built based on the notion of *within-cluster common neighbors* and *inter-cluster common neighbors*. For a given graph $G = (V, E)$, and nodes $u, v, w \in V$, w is said to be a within-cluster common neighbor of u and v if w belongs to the same community as them. Otherwise, w is said to be an inter-cluster common neighbor of u and v. The WIC metric is defined to be the ratio between the size of the set of within- and inter-cluster common neighbors [31].

Running the WIC similarity metric on edges with weight $w(u_1, u_2) \in [0, 1]$ results in this set of edges being reduced in size. Some of its edges are converted to edges with weight $w(u_1, u_2) > 1$ while the remaining are converted to edges with weight $w(u_1, u_2) = 0$, thus increasing the size of the set of attack edges. We terminate this preprocessing with a tuning that aims to scale down all weights $w(u_1, u_2) > 1$ to $w(u_1, u_2) = 1$. The benefit of this transformation is a gain in the accuracy and the stability of the detection mechanism. We are now ready to proceed to the ranking of nodes in order to declare which ones are Sybil nodes, and which ones are benign nodes.

4.3 Trust Propagation

To rank the nodes, each node in the OSN is assigned a *degree-normalized landing probability* of a modified short random walk. The walk starts from a known non-Sybil node. Using this node, we compute the probability of a modified random walk to land on each node u_i after k steps. This landing probability is analogous to the strength of the trust relationship between the nodes, and each step of the walk's probability distribution is considered as a trust propagation process [6].

Early terminated walk: The modified random walk used by SybilRadar is called a short walk because it is an early terminated walk [36]. A random walk that is run long enough will end up with with all the nodes in the social graph having an uniform trust value. The uniform trust value is called the convergence value of the random walk [2]. The number of steps k required for a random walk to converge is called the mixing time of the social graph. Several researches [10,22,24] have shown that for various social networks, the mixing time is larger than $O(\log n)$ with n being the number of nodes in the social graph. To compute the trust values, SybilRadar adapts the Power Iteration method [19]. In SybilRadar the modified power iteration is terminated after $O(logn)$ iterations.

Our modified power iteration method takes as input the transition matrix of social graph, where each element of the matrix is the probability of the random walk to transition from one node to another. The method is executed as a succession of transition matrix multiplications, and at each step the iteration computes the trust distribution over nodes. It works as follows. We define the trust value on a node v after i iterations as $T(v)$, and the total trust as the value

$T \geq 1$. Given $s_1, \ldots, s_k$ the selected trust seeds, we initialize the power iteration by distributing the total trust among the trust seeds as follows:

$$T^{(0)}(v) = \begin{cases} T/k & \text{if } v \text{ is a trusted seed} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

After the initialization step, each node v_i is assigned a trust value $T(v_i)$. The process then proceeds with each node v_i evenly distributing its trust value $T(v_i)$ to each of its neighbor v_j during each round of power iteration. Each node v_i then updates its trust value in accordance with the trust values received from its neighbors. The trust distribution is done proportionally to $w(v_i, v_j) \div deg(v_j)$ which is the ratio of the weight on the edge between the node v_i and its neighbor v_j over the degree of the neighbor node v_j. The use of the weight ensures that a big fraction of the total trust will be distributed to benign accounts rather to Sybil accounts. This results in benign accounts having higher trust value than Sybil accounts. The entire process is summarized in Eq. (4).

$$T^{(k)}(v_i) = \sum_{(v_i, v_j) \in E} T^{(k-1)}(v_j) \frac{w(v_i, v_j)}{deg(v_j)} \tag{4}$$

After $O(logn)$ iterations, the resulting trust value $T(v_i)$ assigned to each node v_i is normalized according to v_i degree. The normalization process involves dividing each node trust value by its degree. This transformation is motivated by the fact that trust propagation is influenced by the node degree, and that this results in the trust propagation being biased toward node with higher degree when the number of iterations grows larger. The normalization ensures that benign nodes get trust values that are close in value [6]. This is influential in identifying Sybil nodes after the ranking.

5 System Evaluation

We first evaluate SybilRadar using both a synthetic network and a real dataset collected from Facebook. For both evaluations we employ procedures that other researchers have used in this line of work. We compare SybilRadar against SybilRank which takes the same structure-based approach that is also based on the use of the power iteration method albeit on an unweighted graph unlike SybilRadar which uses a weighted graph.

Comparing SibilRadar to SybilRank will help highlight the role played by similarity metrics in detecting Sybil accounts. In addition, SybilRank has been demonstrated to outperform other previous structure-based methods [6]. Although Integro outperforms SybilRank, it is not a pure structure-based approach since it leverages account's feature information collected from recent users activities. We have indicated earlier our reservations for using user attributes or activities in Sybil detection. For this reason, we are not including it in our comparison.

Evaluation metric: To express SybilRadar's performance, we use the Area Under the Receiver Operating Characteristic Curve (AUC). AUC for our purpose is defined as the probability to have a randomly selected benign node ranked higher than a randomly selected Sybil node. The AUC is a tradeoff between the False Positive Rate and the True Positive Rate of the classifier. A perfect classifier has an AUC of 1 while a random classifier has an AUC of 0.5. Therefore, we expect our classifier to perform better than a random classifier, and to have an AUC as close as possible to 1.

5.1 Evaluation on Synthetic Networks

The synthetic network is generated using known social network models. First, the honest and the Sybil regions are generated by providing relevant parameters to the network model, like the number of nodes, and the average degree of nodes. Then, the attack edges are generated following the scenario chosen in the experiment. They can be randomly generated or generated in a way to target some specific honest nodes.

Initial Evaluation: We generate the honest region and the Sybil region using the Powerlaw model. The honest region has a size 4000 nodes while the Sybil region has 400 nodes. Both regions have an average degree of 10. The attack scenario chosen simulates an attacker randomly generating 2000 attack edges. The weights on the edges are set to be the values resulting from the two similarity metrics previously described in this Sect. 4.2. For this experiment, we select 20 trust seeds from the honest region. These are supposed to be some nodes that the OSN system administrator is absolutely certain to be honest nodes.

Results: Comparing the ranking quality of both SybilRank and SybilRadar under the chosen scenario, the results show that SybilRadar outperforms SybilRank. SybilRadar resulted in an AUC which is always greater than 0.95, an AUC that is higher than SybilRank's AUC of 0.90.

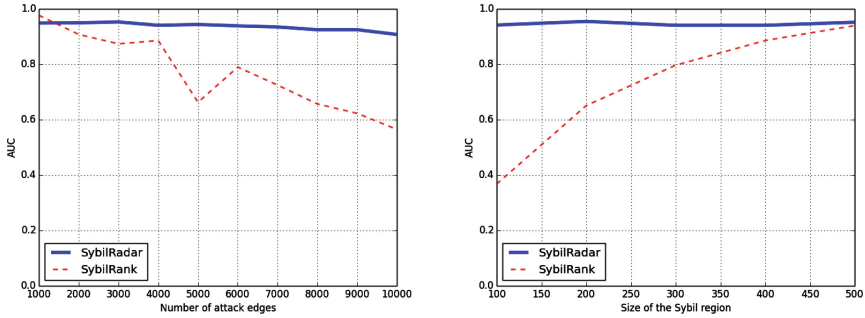
Varying the number of attack edges: In the next experiment, we keep the honest and the Sybil regions as set up in the previous Basic Evaluation. In order to stress-test the platforms being compared, we decide to successively vary the number of attack edges from 1000 to 10000. We want to investigate how the increase in number of attack edges affects the performance of both platforms.

Results: This result can be seen in Fig. 1(a). As the number of attack edges increases, we notice that SybilRank is unable to keep its initial performance, with its AUC dropping from 0.97 to less than 0.6. Meanwhile, the increase in the number of attack edges affects the performance of SybilRadar only marginally. Its AUC still stays above 0.90. This highlights the effectiveness of using similarity metrics in detecting Sybil nodes in the case of social graphs with weak trust.

Varying the size of the Sybil region: In this experiment, we explore how the increase in the size of the Sybil region affects the performance of both platforms.

For this purpose, we design a honest region with 4000 nodes, and an average degree of 10. The attacker is able to create randomly 4000 attack edges. We vary the size of the Sybil region from 100 to 500 nodes each with an average degree of 10.

Results: The experiment results (see Fig. 1(b)) show that SybilRadar and SybilRank react differently to the increase in the size of the Sybil region. When the size of the Sybil region is relatively small compared to the size of the honest region, SybilRank performs poorly. SybilRank performance improves when the size of the Sybil region get relatively bigger. However, as illustrated in Fig. 1(b), SybilRadar displays a stable performance that is less sensitive to the size of the Sybil region.



(a) Varying number of attack edges (b) Varying size of the Sybil region

Fig. 1. Performance on synthetic data

5.2 Evaluation on Real-World Twitter Network

To study if our choice of data in the previous experiments biased our results, we also evaluated the performance of SybilRadar under larger datasets from a different OSN, namely, the Twitter network. The dataset we used is a combination of four datasets: The FakeProject dataset, the Elezioni2013 dataset, the TWT dataset, and the INT dataset [8]. The FakeProject dataset contained profiles of real users who received friend requests from @TheFakeProject, an account created for The FakeProject that was initiated in 2012 at IIT-CNR, in Pisa-Italy. The Elezioni2013 dataset was generated in 2013 for a sociological research undertaken by the University of Perugia and University of Rome, La Sapienza. The TWT dataset and the INT dataset were a set of fake accounts purchased respectively from the fake accounts providers http://twittertechnology.com and http://intertwitter.com. The first two datasets mentioned provided the honest nodes while the last two datasets provided the fake nodes [8].

Pre-processing: Since the Twitter network is directed, we considered only the set of bidirectional edges. This provided us with an initial network of 469,506 nodes and 2,153,427 edges. We further refined this network by removing all nodes with

degree less than 1. The resulting twitter network then comprised 135,942 nodes and 1,819,864 edges. The honest region comprised 100,276 nodes and 634,127 edges while the Sybil region was constituted of 35,666 nodes and 1,086,352 edges. The two regions were connected by 99,385 attack edges.

Results: We ran SybilRadar several times using the Twitter dataset described above. SybilRadar resulted in an AUC which was always greater than 0.95 as shown in Fig. 2.

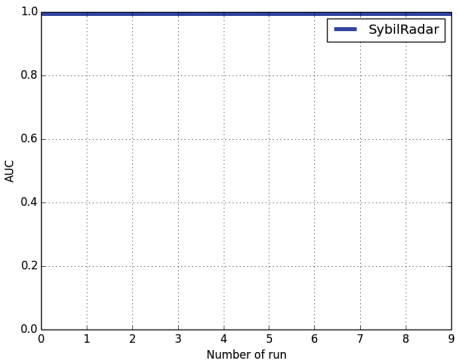


Fig. 2. Performance on Twitter dataset

6 Conclusion

In this paper, we presented a new framework for detecting Sybil attacks in an Online Social Network. In a Sybil attack, an adversary creates a large number of fake identities in an OSN or forges existing identities. The adversary then uses these fake identities to influence the underlying trust basis of the OSN and perform malicious activities such as social spamming, malware distribution and private data collection. Sybils are a significant threat to the OSN. While they cannot be prevented in most OSNs because of their open nature, this work provides a solution by which the OSN operator can automatically, speedily and accurately detect such Sybils.

SybilRadar belongs to the class of Sybil detection techniques that rely on the graph structure of the OSN. This is in contrast to the alternate group of detection mechanisms that rely of identifying features related to user attributes and activities. We believe that while the second class of detection algorithms may provide good detection results on carefully cleaned up OSN data, in real life such data is difficult to obtain since OSN users frequently leave their profiles incomplete or use misleading information purposefully. Moreover, trying to obtain user activity related data may raise serious privacy concerns. As a result, SybilRadar relies on just the structural properties of the OSN graph. We used a variety of OSN test data – both synthetic as well as real-world – to evaluate the detection accuracy of SybilRadar. Our experimental results show that

SybilRadar performs very well – much better than the most well known similar technique – even for OSNs that have the weak trust model and which have a very large number of attack edges between Sybil nodes and honest nodes.

For future work, we plan to add a temporal dimension to our detection framework. Sybil behavior will most likely not be static but change with time. We expect to see major differences in how structural properties of honest nodes change over time and how that of Sybil nodes change. We would like to investigate how this can be modeled to detect Sybils. Also, although we are not a big supporter of using user attributes and activities in Sybil detection, we admit that these techniques can provide somewhat better results. We would like to investigate if and how these techniques can be integrated with SybilRadar so as to improve it but in a manner that does not raise any privacy issues related to OSN users.

References

1. Adamic, L., Adar, E.: Friends and neighbors on the web. Soc. Netw. **25**, 211–230 (2001)
2. Behrends, E.: Introduction to Markov Chains with Special Emphasis on Rapid Mixing. Advanced Lectures in Mathematics. Springer, Wiesbaden (2000)
3. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exp. **2008**(10), P10008 (2008)
4. Boshmaf, Y., Logothetis, D., Siganos, G., Lería, J., Lorenzo, J., Ripeanu, M., Beznosov, K.: Integro: leveraging victim prediction for robust fake account detection in OSNs. In: Proceedings of NDSS (2015)
5. Boshmaf, Y., Muslukhov, I., Beznosov, K., Ripeanu, M.: The socialbot network: when bots socialize for fame and money. In: Proceedings of the 27th Annual Computer Security Applications Conference, pp. 93–102. ACM (2011)
6. Cao, Q., Sirivianos, M., Yang, X., Pregueiro, T.: Aiding the detection of fake accounts in large scale social online services. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, p. 15 USENIX Association (2012)
7. Chang, W., Wu, J.: A Survey of Sybil Attacks in Networks. Technical report, Department of Computer and Information Science, Temple University
8. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: Fame for sale: efficient detection of fake twitter followers. Decis. Support Syst. **80**, 56–71 (2015)
9. Danezis, G., Mittal, P.: Sybilinfer: detecting sybil nodes using social networks. In: NDSS. San Diego, CA (2009)
10. Dellamico, M., Roudier, Y.: A measurement of mixing time in social networks. In: Proceedings of the 5th International Workshop on Security and Trust Management, Saint Malo, France, September 2009
11. Ellison, N.B., et al.: Social network sites: definition, history, and scholarship. J. Comput. Mediat. Commun. **13**(1), 210–230 (2007)
12. Feng, X., Zhao, J., Xu, K.: Link prediction in complex networks: a clustering perspective. Eur. Phys. J. B **85**(1), 1–9 (2012)
13. Fire, M., Tenenboim-Chekina, L., Puzis, R., Lesser, O., Rokach, L., Elovici, Y.: Computationally efficient link prediction in a variety of social networks. ACM Trans. Intell. Syst. Technol. (TIST) **5**(1), 10 (2013)

14. Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., Zhao, B.Y.: Detecting and characterizing social spam campaigns. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 35–47. ACM (2010)
15. Gao, P., Gong, N.Z., Kulkarni, S., Thomas, K., Mittal, P.: Sybilframe: A Defense-in-Depth Framework for Structure-Based Sybil Detection. arXiv preprint. (2015). arXiv:1503.02985
16. Geisser, S.: Predictive Inference, vol. 55. CRC Press, Boca Raton (1993)
17. Ghosh, S., Viswanath, B., Kooti, F., Sharma, N.K., Korlam, G., Benevenuto, F., Ganguly, N., Gummadi, K.P.: Understanding and combating link farming in the twitter social network. In: Proceedings of the 21st International Conference on World Wide Web, pp. 61–70. ACM (2012)
18. Golbeck, J.: Trust and nuanced profile similarity in online social networks. ACM Trans. Web (TWEB) **3**(4), 12 (2009)
19. Golub, G.H., Van der Vorst, H.A.: Eigenvalue computation in the 20th century. J. Comput. Appl. Math. **123**(1), 35–65 (2000)
20. Grier, C., Thomas, K., Paxson, V., Zhang, M.: @ Spam: The underground on 140 characters or less. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 27–37. ACM (2010)
21. Kahanda, I., Neville, J.: Using transactional information to predict link strength in online social networks. ICWSM **9**, 74–81 (2009)
22. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Math. **6**(1), 29–123 (2009)
23. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 243–252. ACM (2010)
24. Mohaisen, A., Yun, A., Kim, Y.: Measuring the mixing time of social graphs. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 383–389. ACM (2010)
25. Nagaraja, S.: Anonymity in the wild: mixes on unstructured networks. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 254–271. Springer, Heidelberg (2007)
26. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (2004)
27. Shi, L., Yu, S., Lou, W., Hou, Y.T.: Sybilshield: An agent-aided social network-based sybil defense among multiple communities. In: Proceedings of IEEE INFOCOM, pp. 1034–1042. IEEE (2013)
28. Stein, T., Chen, E., Mangla, K.: Facebook immune system. In: Proceedings of the 4th Workshop on Social Network Systems, p. 8. ACM (2011)
29. Stringhini, G., Kruegel, C., Vigna, G.: Detecting spammers on social networks. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 1–9. ACM (2010)
30. Thomas, K., Grier, C., Song, D., Paxson, V.: Suspended accounts in retrospect: an analysis of twitter spam. In: Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference, pp. 243–258. ACM (2011)
31. Valverde-Rebaza, J.C., de Andrade Lopes, A.: Link prediction in complex networks based on cluster information. In: Barros, L.N., Finger, M., Pozo, A.T., Gimenénez-Lugo, G.A., Castilho, M. (eds.) SBIA 2012. LNCS, vol. 7589, pp. 92–101. Springer, Heidelberg (2012)

32. Wang, A.H.: Don't follow me: spam detection in twitter. In: Proceedings of the International Conference on Security and Cryptography (SECRYPT), pp. 1–10. IEEE (2010)
33. Yan, G., Chen, G., Eidenbenz, S., Li, N.: Malware propagation in online social networks: nature, dynamics, and defense implications. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 196–206. ACM (2011)
34. Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B.Y., Dai, Y.: Uncovering social network sybils in the wild. ACM Trans. Knowl. Discov. Data (TKDD) **8**(1), 2 (2014)
35. Yao, F., Chen, L.: Similarity propagation based link prediction in bipartite networks. In: Network Security and Communication Engineering: Proceedings of the International Conference on Network Security and Communication Engineering (NSCE 2014), Hong Kong, 25–26 December 2014, p. 295. CRC Press (2015)
36. Yu, H.: Sybil defenses via social networks: a tutorial and survey. ACM SIGACT News **42**(3), 80–101 (2011)
37. Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F.: Sybillimit: A near-optimal social network defense against sybil attacks. In: IEEE Symposium on Security and Privacy, SP 2008, pp. 3–17. IEEE (2008)
38. Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.D.: Sybilguard: Defending against sybil attacks via social networks. IEEE/ACM Trans. Netw. **16**(3), 576–589 (2008)
39. Zhao, X., Li, L., Xue, G.: Authenticating strangers in fast mixing online social networks. In: Global Telecommunications Conference (GLOBECOM 2011), pp. 1–5. IEEE (2011)

Collateral Damage of Facebook Apps: Friends, Providers, and Privacy Interdependence

Iraklis Symeonidis[1(✉)], Fatemeh Shirazi[1], Gergely Biczók[2], Cristina Pérez-Solà[1,3], and Bart Preneel[1]

[1] ESAT/COSIC and iMinds, KU Leuven, Leuven, Belgium
iraklis.symeonidis@esat.kuleuven.be
[2] MTA-BME Future Internet RG,
Budapest University of Technology and Economics, Budapest, Hungary
[3] dEIC, Universitat Autònoma de Barcelona, Barcelona, Spain

Abstract. Third-party apps enable a personalized experience on social networking platforms; however, they give rise to privacy interdependence issues. Apps installed by a user's friends can collect and potentially misuse her personal data inflicting *collateral damage* on the user while leaving her without proper means of control. In this paper, we present a multi-faceted study on the *collateral information collection* of apps in social networks. We conduct a user survey and show that Facebook users are concerned about this issue and the lack of mechanisms to control it. Based on real data, we compute the likelihood of *collateral information collection* affecting users; we show that the probability is significant and depends on both the friendship network and the popularity of the app. We also show its significance by computing the proportion of exposed user attributes including the case of profiling, when several apps are offered by the same provider. Finally, we propose a privacy dashboard concept enabling users to control the *collateral damage*.

1 Introduction

Online Social Networks (OSNs) have become a dominant platform for people to express themselves, interact with each other and get their daily entertainment. By design and popularity, Facebook has morphed into a massive information repository storing users' personal data and logging their interaction with friends, group, events, and pages. The sheer amount and potentially sensitive nature of such data have raised a plethora of privacy issues for Facebook users, such as the lack of user awareness, cumbersome privacy controls, accidental information disclosure, unwanted stalking, and reconstruction of users identities, see Wang et al. [22].

Applications, Providers, Permissions, and Control. Complicating the Facebook privacy landscape, users can also enjoy apps for a personalized social

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 194–208, 2016.
DOI: 10.1007/978-3-319-33630-5_14

experience. Apps can be developed either by Facebook itself or by third-party app Providers (appPs). Facebook relies on permission-based platform security and applies the least privilege principle to third-party apps. For installation and operation, each app requests from the user a set of *permissions*, granting the app the right to access and collect additional information (steps 1 to 4 in Fig. 1a). After the user's approval, apps can collect the user's personal data and store it on servers outside Facebook's ecosystem and completely out of the user's control (steps 5 to 6).

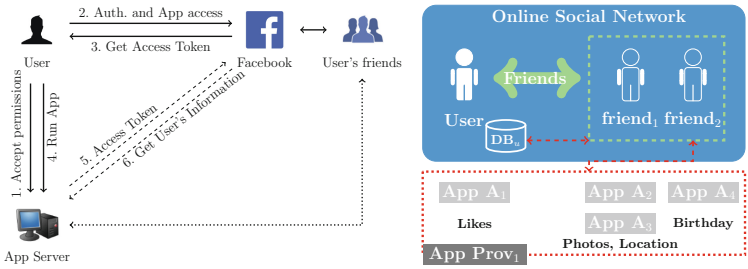


Fig. 1. a. Facebook app architecture, b. *Collateral information collection*

Initially, Facebook enabled apps to collect profile attributes of users' friends by assigning separate permissions to each profile attribute. Later, Facebook has replaced this with a single permission to conform with US Federal Trade Commission (FTC) regulations on data collection [3]. Conformity notwithstanding, apps are still able to collect up to fourteen profile attributes via friends [20]. Of course, users have app-related privacy controls at their disposal; however, they are scattered at multiple locations, such as the user's personal profile (visibility levels per attribute) or the apps menu (attributes friends can bring with them to apps). Taking into account that default settings are very much pro-sharing, fragmented and sometimes curiously worded, privacy control settings could promote incorrectly set policies or complete neglect from users [22].

Privacy Interdependence, Profiling, and Legislation. The suboptimal privacy controls and the server-to-server (and potentially offline) communication between Facebook and appP make any protection mechanism hard to apply [9]. As a result, the user's profile items can be arbitrarily retrieved by an appP without automatic notification or on-demand approval by the user through friends. Since the privacy of an individual user is affected by the decisions of other users (being partly out of their control), this phenomenon is referred to as *privacy interdependence* [6]. From an economic point of view, sharing a user's information without her direct consent can lead to the emergence of externalities. While sharing someone else's information may yield benefits for her (positive externality, e.g., personalized experience in social apps), it is also almost certain to cause a decrease in her utility (negative externality, e.g., exposed profile items). Existing research is limited to pointing out the existence of and risks

stemming from such negative externalities in the Facebook app ecosystem [6], and its potential impact on app adoption [16,17].

Neglected by previous work, third party appPs can be owners of several apps (e.g., appP$_1$ offers app A_1, A_2, A_3 and A_4, see Fig. 1b). For instance, Vipo Komunikacijos and Telaxo are appPs offering 163 and 130 apps, among those 99 and 118 with more than 10,000 monthly active users, respectively (extracted from the Appinspect dataset [5]). As a consequence, an appP may cluster several apps and thus get access to more profile items. Moreover, every app retrieves the Facebook user ID that uniquely identifies a user over apps; hence, the appP could build a combined full profile of the user. We refer to this process as *profiling*, analogously to the term used in the context of consumer behavior in marketing [13]. However, with the help of apps installed by a user's friends, appPs could profile a user partly or entirely without her consent, which constitutes a privacy breach, and could induce legal consequences.

From the legal point of view, both the European Data Protection Directive [2] and the guidelines of FTC [3] require prior user consent for the collection and usage of personal data by data controllers (i.e., Facebook or appPs). According to FTC, apps cannot imply indirect consent through privacy settings; while the European Commission requires transparency and fairness from the data controller about the nature, amount, and aim of data collection: this requirement is not met here with data processing potentially going beyond the users' legitimate expectation.

Motivated by the above privacy issues of Facebook apps we define as *collateral damage* the privacy loss inflicted by the acquisition of users' personal data by apps installed by users' friends, and by appPs offering multiple apps thereby enabling user profiling.

Contribution. We have identified four research questions to further our understanding of indirect and *collateral information collection* in the case of Facebook apps.

- *Are the users aware of and concerned about their friends being able to share their personal data?* We conducted an online survey of 114 participants, to identify the users' views on *collateral information collection, lack of notification and not being asked for their approval.* Our survey provides evidence that participants are very concerned and their concern is bidirectional: the large majority of users wants to be notified and potentially restrict apps' access to profile items both when their friends might leak information about them and vice versa.
- *What is the likelihood that an installed app enables the collateral information collection?* We develop a formula to estimate the probability of this event. We show how the likelihood depends on the number of friends and the number of active users of apps. Moreover, based on results obtained from simulations, we show how the likelihood depends on specific network topologies and app adoption models.

– *How significant is the collateral damage?* We develop a mathematical model and quantify the proportion of user attributes collected by apps installed only by the user's friends, including the case of *profiling*, when several apps belong to the same appP. We compute the significance on several snapshots of the most popular Facebook apps using the Appinspect dataset [5].
– *How can we raise user awareness and help them make informed decisions?* For this end, we discuss a dashboard that enhances transparency by providing an overview of installed apps and the type and total amount of profile attributes collected by apps and, more importantly, appPs.

The rest of the paper is organized as follows. Section 2 presents the user survey. Section 3 presents the mathematical model of *collateral damage* and calculates the likelihood of a user being affected by *collateral information collection*. Section 4 extends the model and quantifies *collateral information collection* illustrated by a case study of popular apps. Section 5 presents the high-level design for a privacy dashboard providing users with proper notifications and control. Section 6 describes future work and concludes the paper.

2 User Survey

In this section, we tackle the research question: "are users concerned about *collateral* information collection?" To answer this question, we conducted an online survey investigating users' views about the disclosure of personal data by Facebook apps installed by the users' friends, and to identify users' concerns about unconsented information collection on Facebook; 114 participants answered the survey. Participants were recruited from the authors' direct and extended friend circles (including mostly, but not only, Facebook friends). Hence, a large proportion of participants have an age between 20 and 35 and are well educated. We found that users are concerned about *collateral information collection* in general, and remarkably concerned when information collection is unconsented. Furthermore, the majority of users prefer to take action to prevent *collateral information collection*. We have to stress that our survey provides us with evidence that users are concerned about the information collection of apps through their users' friends. However, we are not able to extrapolate our findings to the general Facebook population.

2.1 Methodology

After a short introduction, our survey consisted of four main parts. First, we assessed users' standpoints and concerns about default privacy settings and the lack of notification for indirect and unconsented information collection. This assessment is necessary to be able to differentiate users who are concerned independent of their intentions to take actions against such practices. The second part of the survey explores what type of personal data on Facebook users find most sensitive. The third part of our survey is twofold: (1) whether users want

to be notified when their friends' apps can collect their personal data or when their installed apps can collect personal data of their friends; (2) which actions users prefer to take in such cases. Users replied the survey questions by marking their responses on a scale of 1 to 5 where 1 stands for "not concerned at all and 5 stands for "extremely concerned"; we also provided a text field where necessary. The fourth part of the survey collects demographics and information regarding the participants' use of Facebook apps.

2.2 Results

For the first part, we observe that for all four statements users show concern (see Fig. 2). For example, 66 % of users are at least very concerned about the default privacy setting of Facebook that allows apps to collect information from the user's friends. Similarly, 77 % of users are at least very concerned about not being notified when their friends enable *collateral information collection* and 67 % for not being notified when one of the user's own apps can collect their friends' information. Finally, 81 % of users are at least very concerned about *collateral information collection* through apps of their friends without their approval. Note that Golbeck et al. [11] have investigated how informed users are regarding the privacy risks of using Facebook apps. Their findings show that users do not always comprehend what type of data is collected by apps even when they have installed the app themselves. Therefore, it is safe to assume incomplete understanding of apps installed by their friends, which is in line with our results. Note that in Fig. 2, there is a slight difference between participants opinion on statement B on the one hand and statements C and D on the other hand for users which are not concerned. This difference might be because statement B is directly related to the users' information loss. Moreover, statement B would burden the user less than C and D, where action by the users is required.

For the second part of our survey, we found that although users are concerned about a number of attributes, the sensitivity is relatively subjective and differs between users. However, it is noteworthy that certain attributes are standing out and have been marked as sensitive by a large proportion of the participants. For example, most of the users identify photos (84 % are at least very concerned),

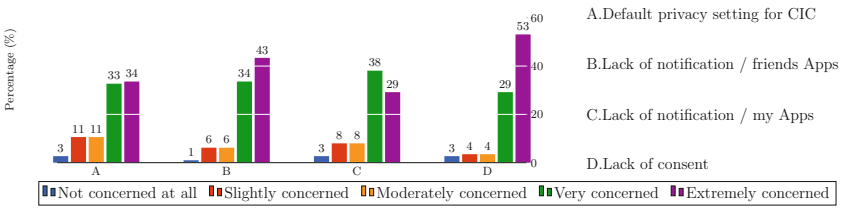


Fig. 2. Results for the first part of the survey where we asked participants about their opinions on four statements regarding default settings, lack of notification (for friends and for the user herself), and lack of consent for *collateral information collection* (CIC).

videos (79 %), their current location (76 %), and family and relationships (54 %) as sensitive profile attributes. The least sensitive profile attributes are proved to be to be birthday and sexual orientation. Note that the sensitivity of the attributes is likely to depend on the context. For example, although a birthday attribute might seem harmless on its own, participants might feel different if a weather app would be collecting this information.

In the third part of the survey, we found that 77 % of users always want to be notified when friends' apps can collect their personal data, 22 % only want to be notified in particular cases, while only about 1 % do not want to be notified at all. Moreover, 69 % of users always want to be notified when their apps are collecting information from their friends, 27 % in particular cases, and only about 1 % not at all. We observe that users are also seriously concerned about damaging their friends' privacy: this corroborates findings on other-regarding preferences from the literature [8,18]. Notification tools can be very useful to enhance privacy awareness for unconsented data collection. Note that Golbeck et al. have shown that the privacy awareness of users can be changed significantly through educational methods [11]. When participants were asked which action they would want to take if notified that friends' apps are about to collect their information (multiple answers allowed), 99 out of 114 participants answered that they would restrict access to their personal data while 8 participants answered that they would unfriend their Facebook friend. Only 5 participants answered that they would take no action. We have to stress that the reaction of a user may strongly depend on the relationship between the user and their friends. When participants were asked what action they would want to take if they are notified that one of their apps is about to collect their friends' information (multiple answers allowed), 64 out of 114 replied that they would restrict access to their friends' personal information for this app. Only 5 out of 114 answered that they would take no action. The answers to the questions in the third part help to confirm that the answers of our participants in the first part were not due to salience bias; participants who were concerned in the first part about not being notified for the *collateral information collection* replied that they also want to take an action in the third part.

The last part of our survey collected demographics and statistics about Facebook and app usage. Participants were between 16 and 53 years old with an average age of 29 years. They have had their Facebook accounts for between 6 months and 10 years, respectively. Moreover, 69 % of our participants have installed an app at least once, and among those 87 % have installed 1 or 2 apps in the last six months. 54 % of the participants were female, 42 % male while 4 % preferred not to disclose their gender. Participants varied greatly in their number of friends, from 10 to 1000. 51 % changed their privacy settings on Facebook; 79 % restricted who could see their profile information, 41 % who could see them in searches, and 35 % who can collect their information through friends apps (multiple answers were allowed). Interestingly, users who already took an action by restricting their permissions to their friends apps by 90 % choose to be notified too. One explanation could be that privacy settings on Facebook are constantly

changing and tracking these changes might be cumbersome [22]. Furthermore, 82 % of our participants had higher education, where 55 % had IT background based on personal interest and 44 % through higher education. We conclude from our survey that users are concerned about the *collateral information collection*, and prefer being notified and try to prevent such type of information collection[1].

3 Likelihood of Collateral Information Collection

In this section, we investigate the likelihood of a user's friend installing an app which enables *collateral information collection*. We build a simple mathematical model and develop a formula to estimate the probability this event occurs. Then, we present case studies taking into account different friendship network topologies and app adoption models. Furthermore, we use the Appinspect dataset [5] to instantiate our estimations, and resort to simulations for computing the probability for different network types.

Let an Online Social Network (OSN) with k users and the corresponding set be denoted by the set $\mathcal{F}$, i.e., $\mathcal{F} = \{u_1, \ldots, u_k\}$. The user is denoted by u, with $u \in \mathcal{F}$. Let f be a friend of u and F^u the set of u's friends, i.e., $f \in \mathsf{F}^u$. Clearly, $\mathsf{F}^u \subseteq \mathcal{F}$. Moreover, let A_j an app and $\mathcal{L}$ the set of all A_js that are offered by the OSN to every u_i, and s the size of the set, i.e., $\mathcal{L} = \{A_1, \ldots, A_s\}$. Moreover, let AU_j be the number of users who have installed A_j. For our likelihood estimation we consider the number of Monthly Active Users (MAU) to represent the number of active users. For instance, currently Facebook has $k = 1.3 \times 10^9$ users (i.e., MAU) [19] and more than $s = 25,000$ Apps [5].

To estimate the likelihood that u's personal data can be collected via the A_j, installed by f, we compute the probability of at least an arbitrary f installing any available A_j. Let Q^f be the probability of f installing A_j which enables *collateral information collection*. For all the friends of u (i.e., F^u) the probability of not installing any A_j is the product of probabilities for each f (this assumes that these probabilities are independent, which seems a reasonable approximation). Let Ω be the probability of at least one of u's friends installing A_j (regardless if u has installed A_j), i.e.,

$$\Omega = 1 - \prod_{f \in \mathsf{F}^u} (1 - Q^f) \ . \tag{1}$$

First, we compute the likelihood Ω when the probability for a friend of the user installing an app is uniformly distributed among all friends.

Case Study 1 – Uniform Distribution. Each f decides whether to install A_j without considering any app adoption signals from other users. The probability of at least a friend of u installing A_j is uniformly distributed among u's friends, and equals $1 - Q$ (Remark: $Q = Q^{f_1} = \cdots = Q^{f_{k'}}$ where $1 \leq k' \leq k$). Q is then computed as all users who installed the app divided by the number of users of the OSN (in the active user sense):

[1] http://iraklissymeonidis.info/survey.

$$Q = \frac{AU_j}{|\mathcal{F}|} \quad . \tag{2}$$

We used the publicly available Appinspect dataset provided by Hubert et al. [5,12] to extract the range of MAU of apps which enable *collateral information collection*. The dataset consists of 16,808 Facebook apps between 2012 and 2014. It contains the application name, id, number of active users (daily, weekly and monthly) and the requested permissions. To illustrate the influence of different values of MAUs on Ω, we consider the upper tier of apps, i.e., over 500,000 MAU, while the most popular app that collects friends' data has 10,000,000 MAU, therefore $5 \cdot 10^5 \leq AU_j \leq 1 \cdot 10^7$. To cover most users, we assume the number of friends for a given u ($|\mathsf{F}^u|$) to be between 0 and 1000. Finally, we estimate the population of Facebook to be $1.1 \cdot 10^9$ MAU for the period of 2012 to 2014 [19].

For A_js with $AU_j \geq 5 \cdot 10^6$ the probability Ω grows steeply with the average number of friends (see Fig. 3a). For an average of 200 friends the probability Ω is more than 0.6. For a user with 300 friends and more, the probability Ω exceeds 0.8. (Note that most Facebook users have more than 200 friends [21].) From Eqs. (1) and (2) it is clear that Ω depends strongly on AU_j. For instance, our most popular app TripAdvisor[2] has approximately $1 \cdot 10^7$ MAU (i.e., $AU_j \approx 1 \cdot 10^7$); assuming that on average a user has 200 friends [21] (i.e., $|\mathsf{F}^u| \approx 200$). Considering $\mathcal{F} = 1.1 \cdot 10^9$ (the population of Facebook) we estimate that the probability of at least one of u's friends installing TripAdvisor is larger than 78 % ($\Omega \geq 0.78$).

Case study 2 – Non-uniform Distribution. Realistic social networks do not conform to the uniformity assumption. Network degree has been reported to follow a power law [15,24] and the clustering coefficient has been found to be much higher than in random networks [15]. Moreover, app adoption has been proclaimed to be affected by different signals [16]. We have resorted to simulations in order to introduce these factors into the estimation of the probability Ω.

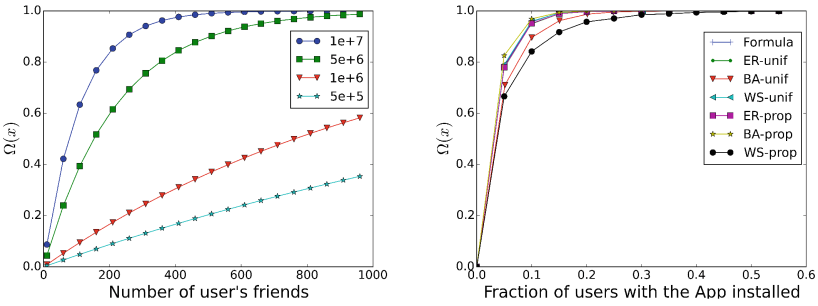


Fig. 3. Likelihood of *collateral information collection* based on a. real data [5] (left, per MAU) and b. simulations (right, with $k = 10,000$ and $d = 30$).

[2] https://www.facebook.com/games/tripadvisor.

Our simulations generate synthetic networks to compute Ω. Regarding the friendship network, we have considered three different, well-known models: Bara- básí-Albert [4] (`BA`), Watts-Strogatz [23] (`WS`), and Erdős-Rényi [10] (`ER`). Regarding app adoption, two different models have been implemented: uniform (`unif`), where all users install an app with the same probability (that is, independently of installations by their friends); and preferential (`prop`), where the probability of a user installing an app is proportional to the number of its friends that have already installed the app.

Regarding the simulations, for each of the configurations (pairs of network and app adoption models), we have computed the probability Ω for one of the user's friends installing an app with respect to the fraction of the users of the network that installed the app. To make the results of different network models comparable, we fixed both the number of nodes in the network, k, and the mean degree, d. Then, we tuned the parameters of the models to achieve these properties.

We performed simulations for network sizes $k \in [100,\ 10,000]$ and mean degree $d \in [10,\ 60]$. Due to space constraints we include the results of just one set of simulations, but the conclusions we have drawn can be extrapolated to the other tested settings. Figure 3b draws the probabilities obtained from networks with $k = 10,000$ and $d = 30$ (results averaged over 100 simulations) and from the analytical uniform app adoption case. Most of the configurations give probability values very close to those obtained when using the formula; the three exceptions are: `ba-unif`, `ws-prop`, and `ba-prop`. The Barabási-Albert model generates graphs with a few very high degree nodes (hubs) and lots of low degree nodes. When combining networks generated with the Barabási-Albert model with a uniform app adoption model, the probability for a hub to install an app is the same as for any other node. To the contrary, when combining `BA` with the `prop` app adoption model, hubs have a higher probability of installing the app than non-hubs, since having a higher degree makes them more likely to have (more) friends with the app installed. As a consequence, each installation affects, in mean, more users, and thus Ω increases. Concerning `ws-prop`, the Watts-Strogatz model generates very clustered networks;[3] when an app is installed by a member of a community, it gets adopted by all other members easily. However, each new installation inside the same community implies a small increase on the overall Ω, because most of the users affected by the installation were already affected by installations from other members of the community. We observe that the probability computation (i.e., Ω) is conditioned on both the network and app adoption models. However, we found that there is a significant probability for a user's friend to install an app which enables *collateral information collection* for different networks and app adoption models.

4 Significance of Collateral Information Collection

In this section, we develop a mathematical model and compute the volume of the user's attributes that can be collected by apps and appPs when installed by

[3] The expected clustering coeff. can be adjusted with the rewiring prob. parameter.

the users' friends. Our calculations are based on several snapshots of the most popular apps on Facebook using the Appinspect dataset [5].

Users and Users' Friends. Each user u_i in an OSN (i.e., $u_i \in \mathcal{F}$) has a personal profile where each u can store, update, delete and administer her personal data [7]. A u's profile consists of attributes a_i such as name, email, birthday and hometown. We denote the set of attributes of a u's profile as $\mathcal{T}$ and n as the size of $\mathcal{T}$, i.e., $\mathcal{T} = \{a_1, \ldots, a_n\}$. For instance, Facebook currently operates with a set of $n = 25$ profile attributes. Let F^{u*} be the union of u's friends and the u itself and f^* an element of F^{u*}, i.e., $f^* \in \mathsf{F}^{u*}$. Clearly, $\mathsf{F}^{u*} = \{u\} \cup \mathsf{F}^u$ and $\mathsf{F}^u \cap \{u\} = \emptyset$, as u is not a friend of u. For instance, $\mathsf{F}^{u*} = \{u, f_1, \ldots, f_{k'}\}$ describes a user u and its k' friends, where $1 \leq k' \leq k$.

Applications and Application Providers. Let $\mathcal{L}$ be the set of apps an app provider (appP) can offer to every u_i in an OSN and s the size of this set, i.e., $\mathcal{L} = \{A_1, \ldots, A_s\}$. Let A_j, for $1 \leq j \leq s$, be the set of attributes that each A_j can collect, i.e., $\mathsf{A}_j \subseteq \mathcal{T}$. Each A_j is owned and managed by an appP denoted by P_j. The set of A_js that belong to P_j it is denoted by P_j, i.e., $\mathsf{P}_j \subseteq \mathcal{L}$. The set of all P_js is denoted by $\mathcal{AP}$ and m the size of the set, i.e., $\mathcal{AP} = \{P_1, \ldots, P_m\}$. From our analysis we identified $s = 16,808$ apps and $m = 2055$ appPs on Facebook indicating that a P_j can have more than one A_j, i.e., $\mathsf{P}_j = \{A_1 \ldots A_{s\prime}\}$ with $1 \leq s\prime \leq 160$ [5].

4.1 Profiling

Application j. When A_j is activated by f^* (i.e., $f^* \in \mathsf{F}^{u*}$), a set of attributes a_i can be collected from u's profile. We denote by $A_j^{u,\mathsf{F}^{u*}}$ an A_j that users in F^{u*} installed and as $\mathsf{A}_j^{u,\mathsf{F}^{u*}}$ the set of attributes a_i that $A_j^{u,\mathsf{F}^{u*}}$ can collect from u's profile. Clearly, $\mathsf{A}_j^{u,\mathsf{F}^{u*}} \subseteq \mathsf{A}_j \subseteq \mathcal{T}$. The set of all $A_j^{u,\mathsf{F}^{u*}}$s installed by the users in F^{u*} is denoted by $\mathsf{L}^{u,\mathsf{F}^{u*}}$. Clearly, $\mathsf{L}^{u,\mathsf{F}^{u*}} \subseteq \mathcal{L}$.

We denote by $\vec{a}_i$ a vector of length n which corresponds to a_i, i.e., $\vec{a}_i = [\overset{1}{0} \ldots \overset{i}{0}1\overset{n}{0}]$. Moreover, we consider $\vec{A}_j^{u,\mathsf{F}^{u*}}$ as a vector of length n, which corresponds to $A_j^{u,\mathsf{F}^{u*}}$, i.e.,

$$\vec{A}_j^{u,\mathsf{F}^{u*}} = \bigvee_{a \in A_j^{u,\mathsf{F}^{u*}}} \vec{a} \Leftrightarrow \vec{A}_j^{u,\mathsf{F}^{u*}}[i] = \begin{cases} 1 & \text{if } a_i \in A_j^{u,\mathsf{F}^{u*}}, \\ 0 & \text{if } a_i \notin A_j^{u,\mathsf{F}^{u*}}, \end{cases} \tag{3}$$

for $1 \leq i \leq n$ and $1 \leq j \leq s$.

Note that:

- $x \cup y = \begin{cases} z = 0 & \text{if } x = y = 0, \\ z = 1 & \text{otherwise.} \end{cases}$ and $\vec{x} \vee \vec{y} = \vec{z}$ where $\vec{x}[i] \vee \vec{y}[i] = \vec{z}[i]$.

For instance, an $\mathsf{A}_j^{u,\mathsf{F}^{u*}} = \{a_1, a_i, a_n\}$ is represented as $\vec{A}_j = \vec{a}_1 \vee \vec{a}_i \vee \vec{a}_n = [\overset{1}{1}0 \ldots \overset{i}{0}1\overset{n}{0}1]$. It represents the attributes that can be collected by A_j when is installed by f (i.e., the user's friend).

Application Provider j**.** Each appP consists of a set of $A_j^{u,\mathsf{F}^{u*}}$s denoted by $\mathsf{P}_j^{u,\mathsf{F}^{u*}}$ which users in F^{u*} installed. Each $\mathsf{P}_j^{u,\mathsf{F}^{u*}}$ can collect attributes of u's profile. To identify which a_is can be collected by P_j we consider $\vec{P}_j^{u,\mathsf{F}^{u*}}$ as a vector of length n (i.e., $n \in \mathcal{T}$), which corresponds to $P_j^{u,\mathsf{F}^{u*}}$, i.e.,

$$\vec{P}_j^{u,\mathsf{F}^{u*}} = \bigvee_{\substack{A \in \mathsf{P}_j^{u,f^*} \\ f^* \in \mathsf{F}^{u*}}} \vec{A}^{u,f^*} = \bigvee_{A \in \mathsf{P}_j^{u,\mathsf{F}^{u*}}} \vec{A}^{u,\mathsf{F}^{u*}} \quad . \tag{4}$$

Note that: $\vec{P}_j^{u,\mathsf{F}^{u*}} = \bigvee_{f^* \in \mathsf{F}^{u*}} \vec{P}_j^{u,f^*} = (\vec{P}_j^{u} \vee \vec{P}_j^{u,f_1} \vee \cdots \vee \vec{P}_j^{u,f_i})$, where $\mathsf{F}^{u*} = \{u, f_1, \ldots, f_i\}$ and $\vec{P}^{u,u} = \vec{P}^{u}$.

The complexity of this operation for all f^* in F^{u*} is $\mathcal{O}(n \times |\mathsf{P}_j^{u,\mathsf{F}^{u*}}|)$.

4.2 Degree of *collateral* Information Collection

Friends f of u ($f \in F^u$) allow access to u's profile by installing A_js. We denote by $\mathit{\Pi}^u_{A_j^u, A_j^{u,\mathsf{F}^u}}$ the number of attributes that can be collected by A_j exclusively from u's friends (and not through the user herself, i.e., $u \notin \mathsf{F}^u$). Let $\vec{\mathit{\Pi}}^u_{A_j^u, A_j^{u,\mathsf{F}^u}}$ be a vector of length n which $\mathit{\Pi}^u_{A_j^u, A_j^{u,\mathsf{F}^u}}$ provides, where $n = |\mathcal{T}|$, where

$$\vec{\mathit{\Pi}}^u_{A_j^u, A_j^{u,\mathsf{F}^u}} = \vec{A}_j'^u \bigwedge \vec{A}_j^{u,\mathsf{F}^u} \quad . \tag{5}$$

Note that: $\vec{x}' \wedge \vec{x} = [\overset{1}{0} \ldots \overset{n}{0}]$ and $\vec{x}' \vee \vec{x} = [\overset{1}{1} \ldots \overset{n}{1}]$.

The complexity of this operation for all f^* in F^{u*} is $\mathcal{O}(n^4 \times |\mathsf{A}_j^u| \times |\mathsf{A}_j^{u,\mathsf{F}^u}|)$.

Similarly, we denote by $\vec{\mathit{\Pi}}^u_{P_j^u, P_j^{u,\mathsf{F}^u}}$ the number of attributes that can be collected by P_j exclusively from u's friends in F^u, i.e.,

$$\vec{\mathit{\Pi}}^u_{P_j^u, P_j^{u,\mathsf{F}^u}} = \vec{P}_j'^u \bigwedge \vec{P}_j^{u,\mathsf{F}^u} \quad . \tag{6}$$

4.3 The Case of Facebook Applications

To examine the problem, we extended our analysis for the apps (i.e., A_js) and appPs (i.e., P_js) on Facebook using the Appinspect dataset [5,12]. For each A_j, apart from the application name and id, the dataset provides us with the requested permissions and the A_js each P_j owns. We computed the proportion of attributes an A_j and P_j can collect through: 1) the user's friends and the user herself (i.e., *profiling*, F^{u*}) and 2) only the user's friends (i.e., *degree of collateral information collection*, F^u). From 16,808 apps, 1202 enables *collateral information collection*. Our analysis focuses on A_js and P_js that have more than 10,000 MAU; there are 207 and 88 respectively in each category[4].

[4] http://iraklissymeonidis.info/Fb_apps_statistics/.

Profiling, F^{u*}. Performing the analysis over the dataset, we found that 72.4 % of A_js and 62.5 % of P_js can collect one attribute from F^{u*}. For all A_js and all P_js, 48.6 % and 28.7 % of attributes which are considered sensitive by the participants of our survey (such as *photos*, *videos*, *location* and *family-relationships*) can be collected. Considering location related attributes such as *current location*, *hometown*, *work_history* and *education_history*, the proportion of attributes that can be collected are 23.5 % from A_js and 23.2 % from P_js.

Degree of Collateral Information Collection, F^{u}. For A_js installed only by F^{u}, 28.9 % of them show a degree of *collateral information collection* equal to 1; similarly, 36.3 % of all P_js. Moreover for F^{u}, we identified that the proportion of sensitive attributes that can be collected from A_js and P_js is 46.8 % and 37 %, respectively; while the proportion of collectable location related attributes is 22.5 % for A_js and 36.9 % for P_js.

We conclude that the size of the two sets of sensitive attributes, collected via profiling versus exclusively through friends, are both significant and, surprisingly, comparable to each other. We also found that a considerable amount of attributes concerning the user's location can be collected by either A_js or P_js.

5 Damage Control: Privacy Dashboard

Our survey results have shown that users from our survey are not only concerned about the *collateral information collection*: they also want to be notified and restrict access to their personal data on Facebook. They also consider removing the apps that can cause *collateral damage*. The need for transparency calls for a Transparency Enhancing Technology solution, raising awareness of personal data collection and supporting the users' decision-making on the sharing of personal data [1,14]. Hence, we propose a dashboard that can help users to manage their privacy more efficiently, and control the *collateral information collection* (see Fig. 4 for an initial user interface design). Technically speaking, the dashboard illustrates how the user's data disclosure takes place through the acquisition of the user's personal data via apps (and respective appPs) installed by their Facebook friends. It displays the nature and proportion of the user's personal data that can be collected by apps and, more importantly, appPs.

From our survey, we have concluded that Facebook users are more concerned about certain types of personal information such as photos, videos, location, and relationships. Our dashboard can accommodate the visualization of *profiling* and the degree of *collateral* information collection by assigning different weights to each attribute in the user's profile. These weights can be then manually fine-tuned by the user. Detailed design and implementation of the dashboard remain the next step in our future work. Additional information such as claimed purpose of collection by the apps can be added in the dashboard. Moreover, further functionality can be added to the dashboard such as leading the users from the dashboard to uninstall the app (this would follow the European Data Protection Directive 95/46/EC [2]).

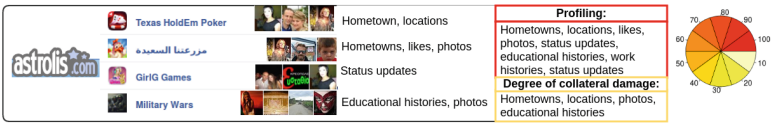


Fig. 4. Privacy dashboard: user interface concept

Finally, our survey shows that users also care about the damage that they might cause to their friends by installing apps (bidirectional concern). Complementing the privacy dashboard, we will also look into providing transparency with an enriched app authorization dialogue at the time of installation. Building on the basic design in [22], the enriched dialogue will direct the attention of users to the existence and volume of *collateral damage* to-be-inflicted on their friends.

6 Conclusion and Future Work

In this paper we have presented a multi-faceted study concerning the *collateral damage* caused by friends' apps in social networking sites. Using a user survey, mathematical modeling, and real data from Facebook, we have demonstrated the importance and quantified the likelihood and significance of such *collateral information collection*. Furthermore, to the best of our knowledge, we have been first to report the potential user *profiling* threat that could be achieved by application providers: they can gain access to complementary subsets of user profile attributes by offering multiple apps.

Our main findings are the following. First, our survey shows that the vast majority of users are very concerned and would like proper notification and control mechanisms regarding information collection by apps installed by their friends. Also, they would potentially like to restrict apps' access to profile items both when their friends' apps might collect information about them and vice versa. As for future work, we are aiming at conducting similar surveys among users of social platforms other than Facebook, and extending the demographic range of participants. We also intend to investigate the relevance of the users concerns and demographic background, attribute values, and sensitivity to particular contexts (e.g., via use cases).

Second, we have quantified the probability that a user is affected by the *collateral information collection* by a friend's app. Assuming a simple app adoption model, an app with more than 500,000 users may indirectly collect information from the average user with 80 % likelihood, irrespective of the user itself having installed the app or not. Moreover, non-uniform app adoption and network models also yield high likelihood. As future work, we aim to extend our simulations regarding both network size and realistic app adoption models.

Third, based on real data, we have quantified the significance of *collateral information collection* by computing the proportion of attributes collected by apps installed by the users' friends. We have found that a significant proportion of sensitive attributes, such as photos, videos, relationships and location, can be

collected from apps either by the user's friends and the user herself (i.e., 48.6 %) or exclusively from the user's friends (i.e., 46.8 %); surprisingly, these values are comparably high. Furthermore, a considerable amount of location-related attributes can be collected by both friends' apps and profiling appPs. As a future work, we aim to enrich our mathematical model by incorporating other parameters such as sensitivity.

Finally, we outline a conceptual design for a privacy dashboard which is able to notify the user about the existence and extent of *collateral damage*, and empower her to take restrictive actions if deemed necessary. We also hint that an enriched app authorization dialogue would complement the dashboard by providing estimates on potential damage to the user's friends at the time of installation. The detailed design and implementation of these solution concepts constitute important future work for us.

Acknowledgments. We notably want to thank Markus Hubert and SBA Research Center for providing us with the necessary material for our study. A thank you to Faruk Gologlu, Filipe Beato, and all the anonymous reviewers who helped for better shaping the idea and the quality of the text. This work was supported in part by the Research Council KU Leuven (C16/15/058), the Spanish Government (TIN2014-55243-P and FPU-AP2010-0078), the Catalan Government (AGAUR 2014SGR-691) and by Microsoft Research through its PhD Scholarship Programme. G. Biczók has been supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

1. Council of the EU Final Compromised Resolution. http://www.europarl.europa.eu. Accessed Feb 2015
2. Directive 95/46/EC of the European Parliament and of the Council. http://ec.europa.eu/. Accessed April 2015
3. FTC and Facebook agreement for 3rd party apps. http://www.ftc.gov/. Accessed February 2015
4. Albert, R., Barabási, A.: Statistical mechanics of complex networks. CoRR, cond-mat/0106096 (2001)
5. AppInspect. A framework for automated security and privacy analysis of OSN application ecosystems. http://ai.sba-research.org/. Accessed Sept 2015
6. Biczók, G., Chia, P.H.: Interdependent privacy: Let me share your data. In 17th FC, Okinawa, Japan, pp. 338–353 (2013)
7. Boyd, D., Ellison, N.B.: Social network sites: definition, history, and scholarship. J. Comput. Mediated Commun. **13**(1), 210–230 (2007)
8. Cooper, D., Kagel, J.H.: Other regarding preferences: a selective survey of experimental results. Handbook of Experimental Economics (2009)
9. Enck, W., Gilbert, P., Chun, B., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.: TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones. Commun. ACM **57**, 99–106 (2014)
10. Erdös, P., Rényi, A.: On the evolution of random graphs. In: Publication of the Mathematical Institute of the Hungarian Academy of Sciences, pp. 17–61 (1960)

11. Golbeck, J., Mauriello, M.L.: User Perception of Facebook App Data Access: A Comparison of Methods and Privacy Concerns. University of Maryland, Maryland (2014)
12. Huber, M., Mulazzani, M., Schrittwieser, S., Weippl, E.R.: Appinspect: large-scale evaluation of social networking apps. In: COSN 2013, Boston, pp. 143–154 (2013)
13. Jobber, D., Ellis-Chadwick, F.: Principles and Practice of Marketing, 7th edn. McGraw-Hill Higher Education, New York (2012)
14. McDonnel, N., Troncoso, C., Tsormpatzoudi, P., Coudert, F., Métayer, L.: Deliverable 5.1: State-of-play: Current practices and solutions. FP7 PRIPARE project. http://pripareproject.eu. Accessed May 2015
15. Mislove, A., Marcon, M., Gummadi, P.K., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: 7th ACM SIGCOMM, San Diego, pp. 29–42 (2007)
16. Pu, Y., Grossklags, J.: An economic model and simulation results of app adoption decisions on networks with interdependent privacy consequences. In: Poovendran, R., Saad, W. (eds.) GameSec 2014. LNCS, vol. 8840, pp. 246–265. Springer, Heidelberg (2014)
17. Pu, Y., Grossklags, J.: Using conjoint analysis to investigate the value of interdependent privacy in social app adoption scenarios. In: 36th ICIS (2015)
18. Stahl, D.O., Haruvy, E.: Other-regarding preferences: Egalitarian warm glow, empathy, and group size. J. Econ. Behav. Organ. **61**, 20–41 (2006)
19. Statista. Leading Social Networks Worldwide as of January 2016. http://www.statista.com. Accessed Sept 2015
20. Symeonidis, I., Tsormpatzoudi, P., Preneel, B.: Collateral damage of online social network applications. In: 2nd ICISSP, Rome (2016)
21. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the Facebook social graph. CoRR, abs/1111.4503 (2011)
22. Wang, N., Xu, H., Grossklags, J.: Third-party apps on Facebook: Privacy and the illusion of control. In: 5th ACM CHIMIT, pp. 4:1–4:10. ACM (2011)
23. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 409–410 (1998)
24. Wilson, C., Boe, B., Sala, A., Puttaswamy, K.P., Zhao, B.Y.: User interactions in social networks and their implications. In: 4th ACM EuroSys, pp. 205–218, New York (2009)

Software Vulnerabilities

Automated Source Code Instrumentation for Verifying Potential Vulnerabilities

Hongzhe Li, Jaesang Oh, Hakjoo Oh, and Heejo Lee(✉)

Department of Computer Science and Engineering, Korea University, Seoul, South Korea
{hongzhe,jaesangoh,hakjoo_oh,heejo}@korea.ac.kr

Abstract. With a rapid yearly growth rate, software vulnerabilities are making great threats to the system safety. In theory, detecting and removing vulnerabilities before the code gets ever deployed can greatly ensure the quality of software released. However, due to the enormous amount of code being developed as well as the lack of human resource and expertise, severe vulnerabilities still remain concealed or cannot be revealed effectively. Current source code auditing tools for vulnerability discovery either generate too many false positives or require overwhelming manual efforts to report actual software flaws. In this paper, we propose an automatic verification mechanism to discover and verify vulnerabilities by using program source instrumentation and concolic testing. In the beginning, we leverage CIL to statically analyze the source code including extracting the program CFG, locating the security sinks and backward tracing the sensitive variables. Subsequently, we perform automated program instrumentation to insert security probes ready for the vulnerability verification. Finally, the instrumented program source is passed to the concolic testing engine to verify and report the existence of an actual vulnerability. We demonstrate the efficacy and efficiency of our mechanism by implementing a prototype system and perform experiments with nearly 4000 test cases from Juliet Test Suite. The results show that our system can verify over 90 % of test cases and it reports buffer overflow flaws with *Precision* = 100 % (0 FP) and *Recall* = 94.91 %. In order to prove the practicability of our system working in real world programs, we also apply our system on 2 popular Linux utilities, Bash and Cpio. As a result, our system finds and verifies vulnerabilities in a fully automatic way with no false positives.

Keywords: Automatic instrumentation · Security sinks · Security constraints · Vulnerability verification

This research was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP)(R0190-15-2011, Development of Vulnerability Discovery Technologies for IoT Software Security).

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 211–226, 2016.
DOI: 10.1007/978-3-319-33630-5_15

1 Introduction

Even though security experts are making best efforts to ensure the software security, the number of software vulnerabilities is still increasing rapidly on a yearly basis, leaving great threats to the safety of software systems. According to the Common Vulnerabilities and Exposures(CVE) database [1], the number of CVE entries has increased from around 1000 CVEs yearly in 2000 to over 8000 yearly in 2015. The discovery and removal of vulnerabilities from software projects have become a critical issue in computer security. Nowadays, because of enormous amount of code being developed as well as limited manpower resource, it becomes harder and harder to audit the entire code and accurately address the target vulnerability.

Security researchers have devoted themselves into developing static analysis tools to find vulnerabilities [9]. The large coverage of code and access to the internal structures makes these approaches very efficient to find potential warnings of vulnerabilities. However, they often approximate or even ignore runtime conditions, which leaves them a great amount of false positives.

Recently, more advanced static analysis methods are proposed [4,5,15]. They either encode insecure coding properties such as missing checks, un-sanitized variables and improper conditions into the analyzer for vulnerability discovery, or they model the known vulnerability properties and generate search patterns to detect unknown vulnerabilities. Even though these approaches can find vulnerabilities using search patterns and exclude the majority of code needed to be inspected, they still require security-specific manual efforts to verify the vulnerability at the very end, which is neither efficient for vulnerability discovery nor feasible for non-security specialists to use it.

According to the previous efforts of researchers, finding exact vulnerabilities in a fully automatic way has been challenging. To automate the overall process of vulnerability detection and verification, we classify the potential vulnerable security sinks into basic 4 types and apply security constraint rules(corresponding to each type) to automatically instrument vulnerability triggering probes into the source code in order to verify vulnerabilities. In this paper, we propose an automatic mechanism to detect and verify software vulnerabilities from C code by using program source instrumentation and concolic testing. In the beginning, we leverage CIL [3] (C intermediate language) to statically analyze the source code including extracting the program CFG(control flow graph), locating the sensitive security sinks and backward tracing the sensitive variables. Subsequently, we classify the security sinks into 4 basic types and perform automated program instrumentation to insert security probes according to different properties of the security sink types, ready for the vulnerability verification. Finally, the instrumented program source is passed to the concolic(CONCrete + symbOLIC) testing engine [10,11] to report and verify the existence of an actual vulnerability. We here focus on buffer overflow vulnerabilities since this type of vulnerability is the major cause for malicious intensions such as invalid memory access, denial of service(system crach) and arbitrary code execution. We demonstrate the efficacy and efficiency of our mechanism by implementing a prototype

system and perform experiments with 4000 buffer overflow test cases from Juliet Test Suite [14]. The results show that our prototype system gets a detection result with $Precision = 100\,\%$ and $Recall = 94.91\,\%$. In order to prove the practicability of our system working in real world programs, we also apply our mechanism on Linux utilities such as Bash and Cpio. As a result, our system finds and verifies vulnerabilities in a fully automatic way with no false positives.

Main contributions of our study are described as follows:

- **Fully Automated Verification for Vulnerability Discovery**. We propose, design and implement a fully automated verification mechanism to detect and verify vulnerabilities with zero interference of manual efforts, which can expand the system usability to more general users such as non-security specialist.
- **Memory Space Analysis(MSA) for Verifying Security Requirements**. We verify the existence of vulnerabilities by generating triggering inputs which violate security constraints($\overline{SC}$). The memory space analysis(MSA) enables us to track the size of buffer space at runtime and set the $\overline{SC}$ conditions accurately. It decreases the false positives for vulnerability verification.

2 Related Work

Source code auditing have been actively researched by security researchers for software assurance and bug finding. Previous researchers have proposed different approaches for static source code auditing and have developed source code static analysis tools for vulnerability discovery.

Flawfinder [9] applies a pattern matching technique to match the security sinks in the source code and report them as vulnerabilities. Even though these approaches can analyze large amount of source code and report vulnerabilities fast, they generate too many false positives due to a lack of analysis about program data flow and control flow information.

Chucky [15] statically taints the source code and identifies missing conditions linked to security sinks to expose missing checks in source code for vulnerability discovery. VCCFinder [4] proposes a method of finding potentially dangerous code with low false positive rate using a combination of code-metric analysis and meta-data from code repositories. It also trains a SVM classifier by the vulnerability commits database to flag code as vulnerable or node. Yamaguchi et al. [5] models the known vulnerability properties and generate search patterns for taint-style vulnerabilities. The generated search patterns are then represented by graph traversals which is used for vulnerability mining in a code property graph database [6]. Even though these approaches can find vulnerabilities using search patterns and exclude the majority of code needed to be inspected, they still require security-specific manual efforts to verify the vulnerability at the very end, which is neither efficient for vulnerability discovery nor feasible for general users(nonspecialist) to use it. Driven by this, there is an urgent need to build

a fully automatic system to accurately catch vulnerabilities within reasonable time as well as the expansion of usability to more general users.

Based on the weakness of the above discussion, we are looking into an automatic and efficient way to do vulnerability verification. Symbolic execution has been proposed to do program path verification but it cannot resolve complex programs with enormous amount of path constraints [12]. Concolic testing [10,11] was proposed to improve symbolic execution in order to make it more practical in real world programs. KLEE [11] was developed to automatically generate high-coverage test cases and to discover deep bugs and security vulnerabilities in a variety of complex code. CREST-BV [10] has shown a better performance than KLEE in branch coverage and the speed of test case generation. However, these approaches suffer from path explosion problem which stops them from scaling to large programs.

CLORIFI [8] is the closest research to this paper. It proposes a method to detect code clone vulnerabilities by the combination of static and dynamic analysis. However, it has not been fully automated and it still requires manual efforts to do source instrumentation for concolic testing, which is still a tedious task. In this paper, we propose, design and implement a fully automated verification mechanism to detect and verify vulnerabilities. In our mechanism, we do vulnerability verification using concolic testing after an automated sink detection and source code instrumentation process, which reduces false positives. We also applies the runtime buffer memory space analysis(MSA) to track the real size of a buffer space which helps us to improve the accuracy of vulnerability discovery.

3 Proposed Mechanism

Discovery of vulnerabilities in a program is a key process to the development of secure systems. In order to find exact vulnerabilities in a fast and accurate way and to reduce the tedious manual efforts for vulnerability verification, we propose a fully automated mechanism to detect and verify software vulnerabilities by taking advantage of both static analysis and concolic testing.

Before we go into detailed description of our approach, the general process is illustrated in Fig. 1. Our mechanism mainly consists of 3 phases which are **code transformation**, **automated instrumentation**, and **vulnerability verification**. In the phase of code transformation, we first leverage the library of CIL to parse the source code into CIL program structures such as function definitions, variables, statements, expressions and so on, and calculate the control flow graph(CFG) information of each function. The reason why we do CIL code transformation is to simplify code structures for efficient static analysis. We then identify the security sinks(potential vulnerable) to get potential vulnerable points. In the second phase, we apply backward data tracing on sensitive variables of each sink to find the variable input location. Then, we perform automatic program instrumentation and prepare the testing object for vulnerability verification. In the last phase of automated instrumentation, we verify each potential security sink to report vulnerabilities using concolic testing.

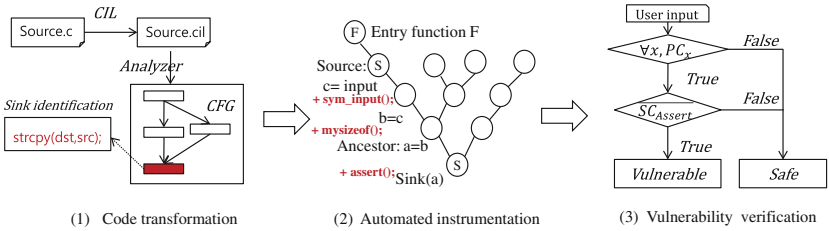


Fig. 1. General overview of our approach.

3.1 Using the CIL

CIL (C Intermediate Language) [3] is a high-level representation along with a set of tools that permit easy analysis and source-to-source transformation of C programs. The reason why we use CIL is that it compiles all valid C programs into a few core constructs without changing the original code semantics. Also CIL has a syntax-directed type system that makes it easy to analyze and manipulate C programs. Moreover, CIL keeps the code location information which enables us to pinpoint the exact location when reporting vulnerabilities(an example is shown in Fig. 5).

By using the provided APIs of CIL, we are able to flatten complex code structures into simple ones (e.g., all looping constructs are reduced to a single form, all function bodies are given explicit return statements). Subsequently, we extract and calculate the control flow graphs(CFGs) of each function using some wrapped module of CIL. Each created node in CFG corresponds to a single instruction in the source code, referred from Sparrow [7], a stable and sound source code analysis framework. Treating the most basic code unit(instruction level) as a CFG node can help us precisely and rapidly address the sensitive sinks and variables as well as providing convenience for the backward sensitive data tracing which will be explained in detail in the following sections.

3.2 Identification of Security Sinks and Sensitive Variables

Since most of buffer overflow vulnerabilities are caused by attacker controlled data falling into a security sink [13], it is crucial to first dig out security sinks. In this paper, we focus on the security sinks which often lead to buffer overflow vulnerabilities. Before that, we explain the definition of security sinks and how they can be classified according to argument properties.

Security Sinks: Sinks are meant to be the points in the flow where data depending from sources is used in a potentially dangerous way. Typical security-sensitive functions and memory access operations are examples of security sinks. Several typical types of security sinks are shown in Table 1.

As we can see from the Table 1, basically, security sinks are either security sensitive functions or a buffer memory assignment operation by index. Further more, according to the number of arguments and whether there is a format

string("%s") argument inside a security sensitive function, we classify the security sensitive functions into 3 types in order to generalize the automatic process of backward tracing and program instrumentation which will be explained in following parts. Along with the last type(buffer assignment), we classify the security sinks of buffer overflow vulnerability into 4 types.

Table 1. Sink classification and variable location

Sink type	Description	Argument format	Sensitive functions	Variable positions
Type 1	functions with two arguments	fn(dst,src)	strcpy, wcscpy strcat, wcscat	2
Type 2	functions with three arguments	fn(dst,src,n)	memcpy, memmove, strncpy strncat, wcsncpy, wcsncat	2,3
Type 3	function with format strings	fn(dst,n,"%s",src)	snprintf, swprintf	2,4
Type 4	memory operations	buffer[i] = expr	dstbuf[i] = 'A'	index:i

- Type 1: Security sensitive functions with 2 arguments: a destination buffer pointer(*dst*) and a source buffer pointer(*src*). The typical argument format is: *fn(dst,src)* and the sensitive variable is the 2nd variable(*src*) in the argument list. The instances of sinks include: *strcpy, wcscpy, strcat and wcscat*.
- Type 2: Security sensitive functions with 3 arguments: a destination buffer pointer(*dst*), a source buffer pointer(*src*) and a number of bytes integer(n). The typical argument format is *fn(dst,src,n)* and sensitive variable is the 2nd variable(*src*) and 3rd argument(n) in the argument list. The instances of sinks include: *memcpy, memmove, strncpy, strncat, wcsncpy and wcsncat*.
- Type 3: The security sensitive functions with format string argument: a destination buffer pointer(*dst*), a number of bytes integer(n), a format string argument and a source buffer pointer(*src*). The typical argument format is *fn(dst,n,format,src)* and the sensitive variable is the 2nd variable(n) and the 4th argument(*src*). The instances of sinks include: *snprintf* and *swprintf*.
- Type 4: The buffers are assigned by some value using buffer array index. This case causes buffer overrun when the index is out of buffer size bound. The typical format is: *buffer[index] = expr* and sensitive variable is the *index*. A instance of this type of sink is: *dstbuf[i] = 'A'*.

After the classification of security sinks, we identify the security sinks as potential vulnerabilities using a fast pattern matching approach over the CIL structures of the source code and extract sensitive variables needed to be backwardly traced in the following step based on the table above.

3.3 Backward Data Tracing

Since instrumentation points are needed before performing automated instrumentation, we propose backward data tracing to find the program input place

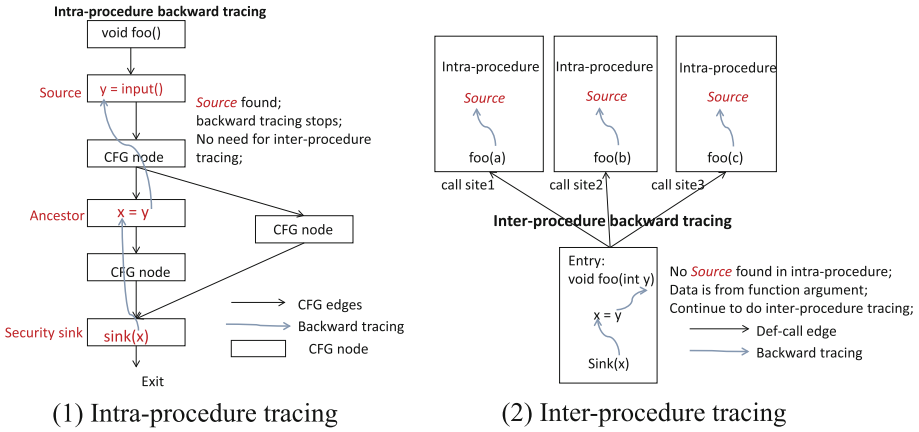


Fig. 2. Backward data tracing.

and treat it as an instrumentation point. Backward data tracing finds the program input location for the corresponding sensitive variables in the sink which reduces the whole input search space of a program. This will help us greatly improve the efficiency for the vulnerability verification. We perform intra- and inter-procedure backward tracing based on the nodes of the control flow graph extracted from CIL. Figure 2 shows the concept of intra-procedure and inter-procedure backward tracing respectively.

Concepts and Definitions. As shown in Fig. 2(1), starting from a security sink, we begin to backwardly trace the corresponding sensitive variable until we reach the source of the sensitive data. In order to understand the process of backward tracing, there are several terms that we need to know.

Source: *Source* is the original definition point of a variable. It is the node where the value of the variable does not depend on any other variable. For instance, the starting points where un-trusted input data is taken by a program. The *Source* is one the following 2 cases.

- $v_0 = gets()$; Assignment node where the left variable is assigned by a user input function such as gets(), scanf() and read(). We also maintain a user input function list.
- $v_0 = 5$; Assignment node where the left variable is assigned by a constant.

Ancestor: The ancestor A of a node N is described as: the traced sensitive data of N gets its value from A. Ancestor nodes are intermediate nodes while sensitive data gets propagated. The ancestor node of a certain variable v_0 could be one of the 4 cases below:

- $v_0 = expression(v_1)$; Node where variable assigned by expression
- $v_0 = f(m, n, ...)$; Node where variable assigned by function return value

– $f(v_0)$; Node where variable assigned inside a function call
– $void\ f(char\ v_0)$; Node for function declaration

The Description of Procedure. As shown in Fig. 2, the intra-procedure backward tracing starts from the initial sensitive variable in the security sink such as $sink(x)$ in Fig. 2(1) and recursively find its ancestor and identify a new variable needed to be traced. The intra-procedure backward tracing stops when the current node is the *Source* of the traced sensitive variable(whole backward tracing also stops) or it stops when the current node is the function entry node. In the later case, *Source* cannot be found in the current function and the data flow comes from argument passing of the function call, so we need further do inter-procedure backward tracing to find the *Source* of the sensitive variable. The inter-procedure tracing(see Fig. 2(2)) starts from intra-procedure tracing. It then checks the return node of intra-procedure backward tracing. If the node is *Source*, the procedure returns and backward tracing ends. If the node is the function entry node, the procedure finds out all the call sites of the current function and identifies the corresponding sensitive variable in each call site. Then it applies intra-procedure backward tracing on sensitive variables in each call site as a new starting point. The whole backward tracing stops and exits when the *Source* of the sensitive variable in the security sink is found.

3.4 Program Source Instrumentation

After backward tracing, we get security sinks and *Sources* of the corresponding sink and store them into a list of sinks(sink_list) and a list of Sources(source_list) accordingly. We also establish a sink-source mapping between the 2 lists which helps us to correctly find the corresponding source for a certain sink. To instrument the program source, we make security probes(assertions) based on our predefined security requirements (Table 2) right before the *security sink* and replace the source input statement with symbolic values. To automate the overall process of source code instrumentation, we generalize the security constraint rules for the 4 basic types of vulnerability to automatically instrument bug triggering probes into the source code in a more generalized way.

Program Constraints(PC) and Security Constraints(SC): Program constraints are generated by following a specific branch according to conditional statements in the program. Program inputs which satisfy a set of program constraints are meant to execute a certain path of the program. Security constraints are clearly high-level security requirements, which is also used as our baseline to make security probes before the security sinks. For instance, the length of the string copied to a buffer must not exceed the capacity of the buffer. We define security requirements for security sinks such us security-sensitive functions and buffer assignment with index based on the condition that related arguments must satisfy to ensure the security of software systems. In our work, we have collected 14 library functions from Juliet Test Suite which are well known to be “insecure” for buffer overflows as security-sensitive functions and generated

security requirements for them. Table 2 shows part of our predefined security constraints for security sinks. When there are inputs which satisfy program constraints but violates security constraints($PC \wedge \overline{SC}$) at a certain point during the execution, the program is considered to be vulnerable. To suggest a way to extend the method to cover other types of vulnerabilities, we will investigate the vulnerability features and define more sinks and the corresponding security requirements for vulnerability types such as integer overflows, format strings and divide-by-zeros.

Table 2. Security requirements for security sinks

Security sinks	Security requirement	Description
strcpy(dst,src)	dst.space > src.strlen	Space of dst should be bigger than the string length of src
strncpy(dst,src,n)	(dst.space ≥ n) ∧ (n ≥ 0)	Space of dst should be bigger or equal to the positive integer n
strcat(dst,src)	dst.space > dst.strln + src.strlen	Space of dst should be bigger than the total string length of dst and src
getcwd(buf,size)	(buf.space ≥ size) ∧ (size ≥ 0)	Space of buf should be bigger or equal to the positive integer size
fgets(buf,size,f)	(dst.space ≥ size) ∧ (size ≥ 0)	Space of dst should be bigger or equal to the positive integer size
buf[i] = expr.	buf.space > i	Space of buf should be bigger than the index value i

Instrument Input Location and the Security Sink: To instrument the input location, we first iterate over all the element in the *source_list*, if the current *Source* takes the data from user input such as command line, network data, or a file, we replace the user input data with symbolic input values. For example, "$a = readfile()$;" will be replaced by "$a = sym\_input()$;".

To instrument the security sink, we insert an assertion statement right before the security sink based on the security rules defined in Table 2. For example, before the sink $strcpy(dst, src)$, we insert an assertion statement $assert(dst.space > strlen(src))$. However, there is a problem here. We can easily get the length of a string by using "$strlen()$" function(C library function), but the space of a buffer is hard to be determined at runtime as well as at compile time. Someone may say, we can always get the size of a buffer by "$sizeof()$". This is not correct when we measure the size of the buffer memory space. E.g., if we want to get the real space that *dst* points to, we cannot use "$sizeof(dst)$"

because it will always return the actual size of a pointer itself which is the number 4 at 32 bit architectures. In order to get the real buffer size of a buffer pointer, we propose a pointer analysis approach to correctly get the buffer size at program runtime.

Memory Space Analysis(MSA) at Runtime: As we can see from Table 2, in the security requirement rules, we have to get the "buffer.space" value to accurately set the bug triggering condition and instrument the sinks. However, the common static analysis of source code cannot get the runtime updating information about memory size of a buffer pointer. This usually will result in an inaccurate assertion of SC violating condition, which makes the system generate possible false positives when reporting vulnerabilities. The runtime memory space analysis enables us to accurately track the real size of a pointer's buffer space and helps us to correctly instrument the program so as to ensure high accuracy of vulnerability verification. We make a library called "libmysizeof" and it provides 3 major functions: *mysizeof_store()*, *mysizeof_propagate()* and *mysizeof_getspace()*. We insert *mysizeof* functions in the corresponding place in the source code. The steps are shown below:

- Iterate over all the instructions in the source code and identify buffer allocation statement such as *buf = malloc(n);* and *char buf[100];*. Then, we store the buffer pointer name and the corresponding size in a global map M by inserting *mysizeof_store()* function at the current location.
- Identify each pointer propagation instruction such us *p_2 = p_1 + 5;*. The *mysizeof_propagate()* will propagate the size of *p_1* to get the size of *p_2* according to the operation and store the size of *p_2* into the map M. We then insert this *mysizeof_propagate()* function at the current location.
- When we need to get the runtime size of a pointer's buffer space, we insert *mysizeof_getspace()* function at a certain location in the source code to get the correct buffer space.

After inserting "mysizeof" functions, we can get the accurate size of a pointer's buffer space at runtime. The buffer size information can then be used in the assertions. For instance, $assert(mysizeof\_getspace(dst) > strlen(src))$. Figure 3 shows an example of instrumenting pointer analysis functions along with input and sink instrumentation.

Until here, we prepare a testing source object from the program input to the potential vulnerable sinks. This object is usually a small part of the whole program source which helps us to release the burden of next stage.

3.5 Vulnerability Verification Using Concolic Testing

We apply concolic testing in our mechanism to verify the potential vulnerabilities. The general principle of the verification is to find inputs which satisfy all the program constraints(PCs) but violate the security constraints(SCs) as shown in Fig. 1. Symbolic execution and concolic execution have been widely used in software testing and some have shown good practical impact, such as KLEE [11] and

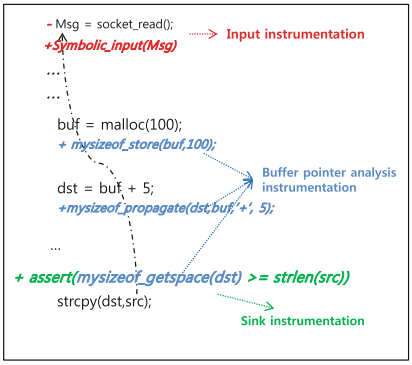


Fig. 3. Automatic instrumentation.

CUTE [2]. However, they suffer from path explosion problem which makes them cannot scale well to large real world programs. In our scenario, the backward tracing module helps us to find the program inputs which are related to the sensitive data instead of the whole program input space. This can mitigate the path explosion problem mentioned before. Our approach for concolic testing to verify potential vulnerabilities mainly follows a general concolic testing procedure [10]. However, the difference is that we focus on generating an input to execute the vulnerable branch instead of generating inputs to traverse every possible paths of the program. Hence, it is more cost efficient when doing concolic testing.

4 Experimental Results

4.1 Implementation

System Prototype: We have implemented our mechanism by developing a prototype system. Our system consists of 3 phases: **code transformation**, **automated instrumentation**, and **vulnerability verification**. Its architecture is described in Fig. 4. The system is used to discover buffer overflow vulnerabilities in software projects.[1]

Environment Setup: We performed all experiments to test our automatic vulnerability detection and verification system on a desktop machine running Linux Ubuntu 14.04 LTS (3.3 GHz Intel Core i7 CPU, 8 GB memory, 512 GB hard drive).

Dataset: For the experiment, we prepared 2 kinds of datasets - Juliet Test Suite and Linux utilities. First one is Juliet Test Suite provided by US National Security Agency(NSA) which has been widely used to test the effectiveness of vulnerability detection tools. To test our tools, we prepared 3,969 files for stack based buffer overflow vulnerabilities, each of which belongs to 4 basic types

[1] Our program and sample scripts are available at http://cssa.korea.ac.kr/clorifi.

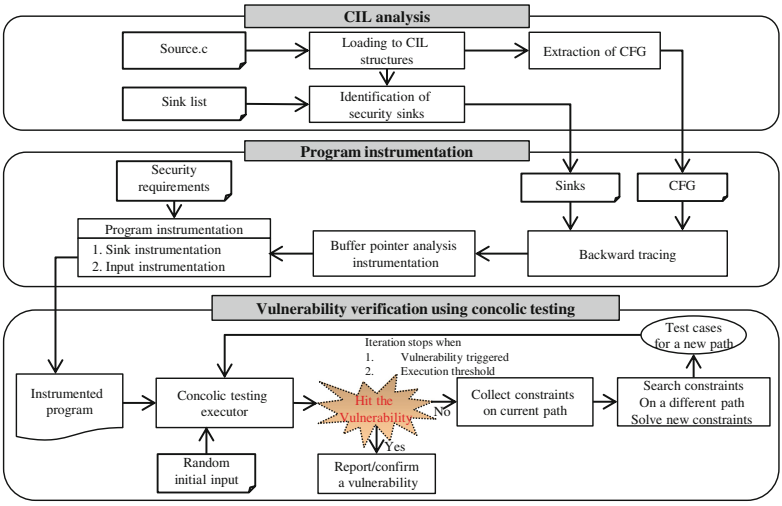


Fig. 4. The system architecture.

based on our sink classification in Table 1. The second dataset is 2 famous Linux utilities, Bash and Cpio, which is used to prove the practicability of our system.

4.2 Experimental Results

We have conducted our experiments with two types of dataset.

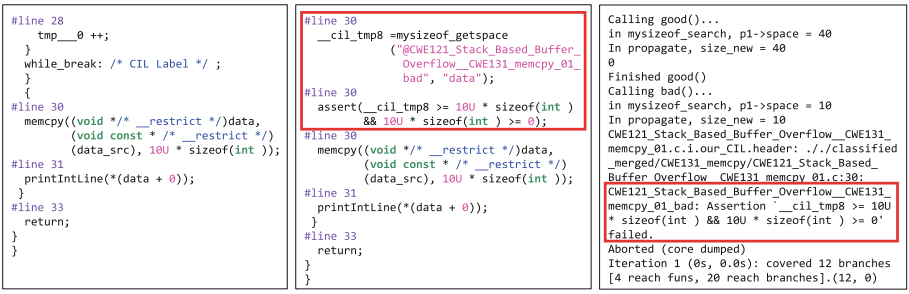
Juliet Test Suite. We tested our approach with 3,969 testcases of Stack Based Buffer Overflow vulnerabilities of Juliet Test Suite. The number of samples covered in Table 3 states the number of cases that we verified (over 90 % in total). Our processing time for 3,969 testcases is nearly 20 min, which includes about 2 min of instrumentation and 17 min of concolic testing. Table 3 shows the number of testcases processed, number of good and bad sinks, and elapsed time of instrumentation and concolic testing for each type.

We checked the rest of the cases that we couldn't verify and found out that our frontend parser cannot handle non-standard coding style such as wide string L"AAA"(222 cases). Besides, our tool failed to correctly instrument the source code when handling function pointers(76 cases) and complex use of pointers such us pointer's pointer(105 cases).

A working example is shown in Fig. 5. Sub figures in Fig. 5 indicate the sequence of our mechanism. Figure 5(1) shows the result of CIL transformation of a file named CWE121_Stack_Based_Buffer_Overflow__CWE131_memcpy_01.c. Then we show the instrumentation result of this case in Fig. 5(2). The assertion $assert(dst.space \geq n) \wedge (n \geq 0)$ is automatically inserted before the sink *memcpy*. Figure 5(3) shows the execution result of concolic testing. In this step, it actually verifies the vulnerability by checking whether the execution of program violates

Table 3. Type of vulnerabilities and base information of experiment

Type	Number of Samples Covered	Elapsed Time (Instrumentation)	Elapsed Time (Concolic Testing)	Number of Bad Sinks	Number of Good Sinks
1	2,065/2,217	78.816 s	547.879 s	2,217	3,126
2	695/732	25.893 s	175.986 s	732	1,008
3	292/296	10.535 s	68.504 s	296	408
4	715/724	26.150 s	197.690 s	724	1,048
Total	3,767/3,969	141.394 s	990.059 s	3,969	5,590



(1) CIL transformation of source code (2) Automated Instrumentation (3) Vulnerability Verification

Fig. 5. Snapshot results of different phases

the assertion or not. By using this mechanism, our approach can detect and verify the vulnerabilities in the Juliet Test Suite in a fully automatic way.

The Comparative Detection Results. We show the number of false positives regarding to each type of sink in Table 4. As we can see, when we apply our system with MSA, we get no false positives while there are some false positives(233 in total) when MSA is not applied. The memory space analysis technique finds the real size of the buffer space at runtime and accurately set the condition for violating security constraint(bug triggering condition). When the MSA is not applied, the real runtime memory size of a pointer in the violation condition can only be set by approximation, which results in false positives. MSA can help our system completely reduce false positives.

We also compare our system with Flawfinder [9] which is a source code auditing tool widely used in security communities. We measure the precision ($\frac{TP}{TP+FP}$), recall($\frac{TP}{TP+FN}$) and F1_Value($\frac{2P*R}{P+R}$) for each tool: (1) our system with MSA; (2) our system without MSA; (3) Flawfinder. As shown in Fig. 6, our system applied with MSA gets the highest *Precision* of 100 % which means 0 false positives. In terms of *Recall*, Our system with MSA gets *Recall* of 94.91 %. Flawfinder has the highest *Recall* value, however, its *Precision* is quite low. *F1_value* is a more comprehensive indicator for evaluation, our system with MSA gets the highest *F1_value* of 97.43 %. For the false negatives, our tool failed to correctly instrument the source code when handling the cases involving function pointers and complex use of pointers such as pointer's pointer(total 181 cases). This makes the

Table 4. False positives with or w/o MSA

Sink type	# of FP with MSA	# of FP without MSA
1	0	102
2	0	58
3	0	25
4	0	48
Total	0	233

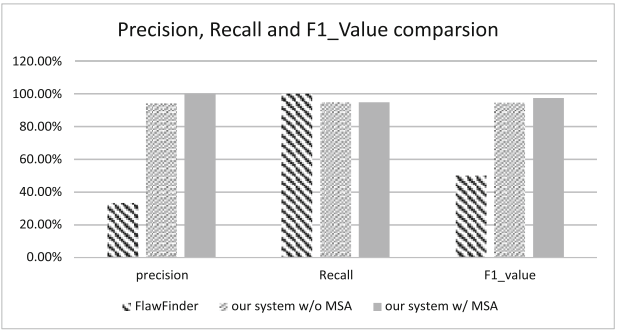


Fig. 6. Detection performance and comparison

tool cannot trigger the failure of the assertion when a real vulnerability exists, which contributes to false negatives.

Case1: Cpio-2.6(CVE-2014-9112). We also demonstrate the effectiveness of our system by real open source projects. Figure 7 shows a vulnerability in program Cpio-2.6 which is a program to manage archives of files. This vulnerability is caused by an integer overflow induced buffer overflow. at line 8, the numeric

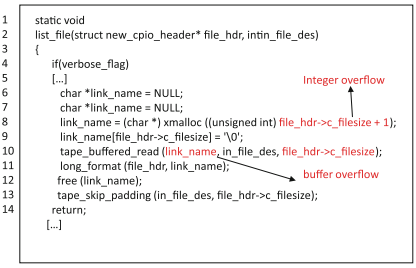


Fig. 7. CVE-2014-9112 vulnerability from Cpio-2.6

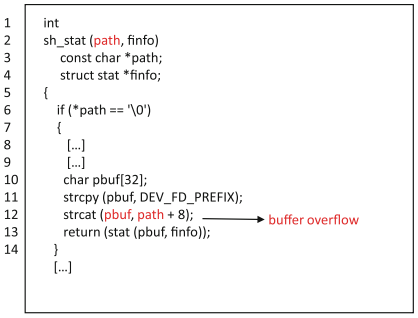


Fig. 8. CVE-2012-3410 vulnerability from Bash-4.2

operation can cause an integer overflow and results in 0 bytes allocation for "link_name". The buffer overflow is at line 10 when the program is trying to write "c_filesize" number of bytes to 0 space buffer. We apply our system to automatically report out this vulnerability in a fully automatic way by generating an input which makes "$filesize\_c = 0xffffffff$".

Case2: Bash-4.2(CVE-2012-3410). We also apply our system to Bash-4.2 and successfully verifies the vulnerability in Fig. 8. Our system identifies the sink "strcat", backwardly tracing the user input and set the violating condition of security constraint by automatic instrumentation. The system reports out this vulnerability with a triggering input "$path = /dev/fd/aaaa...aa(35'a's)$".

5 Conclusion

In this paper, we propose, design and implement an automated verification mechanism for vulnerability discovery. Different from other source code auditing methods, our mechanism needs no human interference with extremely low false positive rate and it can expand the system usability to non-security specialist. It takes source code as input, detects and verifies the existence of vulnerability in a fully automatic way. What's more, the memory space analysis(MSA) enables us to set violating condition of security requirements accurately so as to report vulnerabilities with higher accuracy. We developed a prototype system and conducted several experiments with Juliet test cases and also real open source projects. The results show that our system can detect and verify vulnerabilities with low false positives within reasonable time.

However, there are concerns and limitations as well. To the current stage, our system focuses on buffer overflow vulnerability. In future research, we will study the features of other kinds of vulnerability and expand the vulnerability type coverage. Moreover, due to the incapability of handling complex data types such as nested structures in C code, function pointers and pointer's pointer, the system is limited to be working on programs with relatively small amount of source code. The source code analysis and automatic instrumentation will be further generalized to cover large programs.

References

1. MITRE group.: Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/
2. Sen, K., Marinov, D., Agha, G.: Cute: a concolic unit testing engine for C. In: ACM International Symposium on Foundations of Software Engineering, pp. 263–272 (2005)
3. Necula, G.C., McPeak, S., Rahul, S.P., Weimer, W.: CIL: Intermediate language and tools for analysis and transformation of C programs. In: Nigel Horspool, R. (ed.) CC 2002. LNCS, vol. 2304, pp. 213–228. Springer, Heidelberg (2002)
4. Perl, H., Dechand, S., Smith, M., Arp, D., Yamaguchi, F., Rieck, K., Acar, Y.: Vccfinder: Finding potential vulnerabilities in open-source projects to assist code audits. In: Proceedings of the 22nd ACM CCS, pp. 426–437 (2015)

5. Yamaguchi, F., Maier, A., Gascon, H., Rieck, K.: Automatic inference of search patterns for taint-style vulnerabilities. In: IEEE Symposium of Security and Privacy, pp. 797–812 (2015)
6. Yamaguchi, F., Golde, N., Arp, D., Rieck, K.: Modeling and discovering vulnerabilities with code property graphs. In: IEEE Symposium of Security and Privacy, pp. 590–604 (2014)
7. Oh, H., Lee, W., Heo, K., Yang, H., Yi, K.: Selective context-sensitivity guided by impact pre-analysis. ACM SIGPLAN Not. **49**(6), 475–484 (2014)
8. Li, H., Kwon, H., Kwon, J., Lee, H.: CLORIFI: software vulnerability discovery using code clone verification. Pract. Experience Concurrency Comput. **28**(6), 1900–1917 (2015)
9. Wheeler, D.: Flawfinder (2011). http://www.dwheeler.com/flawfinder
10. Kim, M., Kim, Y., Jang, Y.: Industrial application of concolic testing on embedded software: Case studies. In: IEEE International Conference on Software Testing, Verification and Validation, pp. 390–399 (2012)
11. Cadar, C., Dunbar, D., Engler, D.: Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: USENIX Symposium on Operating Systems Design and Implementation, vol. 8, pp. 209–224 (2008)
12. Zhang, D., Liu, D., Lei, Y., Kung, D., Csallner, C., Wang, W.: Detecting vulnerabilities in C programs using trace-based testing. In: IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 241–250 (2010)
13. Di Paola, S.: Sinks: Dom Xss Test Cases Wiki Project. http://code.google.com/p/domxsswiki/wiki/Sinks
14. Boland, T., Black, P.E.: Juliet 1.1 C/C++ and Java test suite. J. Comput. **45**(10), 89–90 (2012)
15. Yamaguchi, F., Wressnegger, C., Gascon, H., Rieck, K.: Chucky: Exposing missing checks in source code for vulnerability discovery. In: ACM CCS, pp. 499–510 (2013)

An Information Flow-Based Taxonomy to Understand the Nature of Software Vulnerabilities

Daniela Oliveira[1(✉)], Jedidiah Crandall[2], Harry Kalodner[3], Nicole Morin[4], Megan Maher[4], Jesus Navarro[5], and Felix Emiliano[4]

[1] University of Florida, Gainesville, USA
daniela@ece.ufl.edu
[2] University of New Mexico, Albuquerque, USA
[3] Princeton University, Princeton, USA
[4] Bowdoin College, Brunswick, USA
[5] NVIDIA, Santa Clara, USA

Abstract. Despite the emphasis on building secure software, the number of vulnerabilities found in our systems is increasing every year, and well-understood vulnerabilities continue to be exploited. A common response to vulnerabilities is patch-based mitigation, which does not completely address the flaw and is often circumvented by an adversary. The problem actually lies in a lack of understanding of the nature of vulnerabilities. Vulnerability taxonomies have been proposed, but their usability is limited because of their ambiguity and complexity. This paper presents a taxonomy that views vulnerabilities as fractures in the interpretation of information as it flows in the system. It also presents a machine learning study validating the taxonomy's unambiguity. A manually labeled set of 641 vulnerabilities trained a classifier that automatically categorized more than 70000 vulnerabilities from three distinct databases with an average success rate of 80 %. Important lessons learned are discussed such as (i) approximately 12 % of the studied reports provide insufficient information about vulnerabilities, and (ii) the roles of the reporter and developer are not leveraged, especially regarding information about tools used to find vulnerabilities and approaches to address them.

1 Introduction

Despite the security community emphasis on the importance of building secure software, the number of new vulnerabilities found in our systems is increasing with time; The 2014 Symantec Internet Security report announced that 6,787 *new* vulnerabilities occurred in 2013. This represents a 28 % increase in the period 2013–2014, compared to a 6 % increase in the period 2012–2013 [5]. Further, old and well-studied vulnerabilities, such as buffer overflows and SQL injections, are still repeatedly reported [3].

This material is based upon work supported by the National Science Foundation under Grant Nos. #1149730, #0844880, #0905177, and #1017602.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 227–242, 2016.
DOI: 10.1007/978-3-319-33630-5_16

A common approach to address vulnerabilities is patch-based mitigation targeting specific exploits. This approach may not completely address the vulnerability since it fails to address its essence, and does not generalize well with similar vulnerabilities exploited differently. Take the file-system TOCTTOU vulnerability as an example. Dean and Hu [17] provided a probabilistic solution for filesystem TOCTTOU that relied on decreasing the chances of an attacker to win all races. In their solution, the invocation of the `access()` ... `open()` sequence of system calls is followed by k additional calls to this pair of system calls. From the application layer viewpoint, the solution addresses the concurrency issue because the chances that the attacker will win all rounds are small. Borisov *et al.* [10], however, observed that this vulnerability crosses the boundary between the application and the operating system layers, and allowed an attacker to win the race by slowing down filesystem operations. This caused the victim process to be likely suspended after a call to `access()`.

A first step towards viewing cyber security as a science is understanding software vulnerabilities scientifically. Weber *et al.* [31] also argue that a good understanding and systematization of vulnerabilities aids the development of static-analysis or model checking tools for automated discovering of security flaws.

Taxonomies decrease the complexity of understanding concepts in a particular field. Taxonomy-based vulnerability studies have been tried since the 70 s [7,8,18,21] but they were proved ambiguous by Bishop and Bailey [9], who showed how the same vulnerability was put into multiple categories depending on the layer of abstraction it was being analyzed. The other problem with current taxonomies is their complexity. For example, CWE v1.9 has 668 weaknesses and 1043 pages. Ambiguous and complex taxonomies not only confuse a developer, but also hinder the widespread development of automated diagnosis tools leveraging its categories as points for checks.

This paper introduces a concise taxonomy for understanding the nature of vulnerabilities that views vulnerabilities as fractures in the interpretation of information as it flows in the system. In a seminal paper on computer viruses [15], Cohen said that "*information only has meaning in that it is subject to interpretation.*" This fact is at the crux of vulnerabilities in systems. As information flows from one process to another and influences the receiving process' behavior, interpretations of that information can lead to the receiving process doing things on the sending process' behalf that the system designer did not intend to allow as per the security model. Information, when viewed from the different perspectives for the various levels of abstraction that make up the system (OS, application, compiler, architecture, Web scripting engine, *etc.*), should still basically have the same interpretation. The lack of understanding on the nature of vulnerabilities cause defense solutions to focus on only one perspective (application, compiler, OS, victim process or attacker process) and become just mitigation solutions that are rapidly circumvented by a knowledgeable adversary.

To validate the unambiguity and usefulness of this taxonomy, a machine learning-based [32] study was conducted using a training set of 641 manually

classified vulnerabilities from three public databases: SecurityFocus [35], National Vulnerability Database (NVD) [1] and Open Sourced Vulnerability Database (OSVDB) [2]. This manually labeled set was used to train a machine learning classifier built with the Weka suite of machine learning software [32]. More than 70000 vulnerabilities from a ten year period from the three databases were automatically classified with an average success rate of 80 %, demonstrating the unambiguity potential of the taxonomy.

Important lessons learned in this study are discussed. First, there are a significant number of poorly reported vulnerabilities (approximately 12 % of the vulnerabilities in the manually classified set), with descriptions containing insufficient or ambiguous information. This type of report pollutes the databases and makes it hard to address vulnerabilities scientifically, and disseminate relevant information to the security community. Second, the roles of the reporter and the developer are not leveraged and important information has not been added to reports, such as tools used to find vulnerabilities and approaches taken to address them. Finally, the lack of standards on vulnerability reports and across databases adds complexity to the goal of addressing vulnerabilities scientifically, as they are viewed as dissimilar, independent and unique objects. The paper also discusses the application of such taxonomy in the context of automated diagnosis tools to assist the developer.

This paper's contributions are as follows:

1. A concise taxonomy for understanding the nature of vulnerabilities based on information-flow that can be easily generalized and understood is proposed.
2. The taxonomy's categories and their information-flow nature are discussed against notorious vulnerabilities, such as buffer overflows, SQL injection, XSS, CSRF, TOCTTOU, side-channels, DoS, *etc.*.
3. A large scale machine learning study validating the taxonomy's unambiguity is presented. In this study a manually labeled set of 641 vulnerabilities trained a classifier that automatically categorized more than 70000 vulnerabilities from three distinct databases with an average success rate of 80 %.
4. Important lessons learned are discussed such as (i) approximately 12 % of the studied reports provide insufficient information about vulnerabilities, and (ii) the roles of the reporter and developer are not leveraged, especially regarding information about tools used to find vulnerabilities and approaches to address them.
5. A discussion of the application of this taxonomy in automated diagnosis tools is provided.

The rest of the paper is organized as follows. Section 2 presents the proposed taxonomy and discusses notorious vulnerabilities from the perspective of information flow. Section 3 presents the machine learning study conducted to evaluate the taxonomy. Section 4 discusses related work and Sect. 5 concludes the paper.

2 The Taxonomy

This paper introduces a new vulnerability taxonomy based on information flow. The goal was to produce an unambiguous taxonomy that can be leveraged to address software vulnerabilities scientifically. Vulnerabilities are viewed as fractures in the interpretation of information as it flows in the system. Table 1 details with examples the proposed taxonomy and its categories. The following sections describe each one of these categories with some examples and how they can be viewed in terms of information flow.

Please notice that there is no *design flaw* category because this study understands that all vulnerabilities are ultimately caused by design flaws. Vulnerabilities are weaknesses in the design and/or implementation of a piece of software that allow an adversary to violate the system security policies regarding the three computer security pillars: confidentiality, integrity and availability.

2.1 Control-Flow Hijacking

These vulnerabilities allow an attacker to craft an exploit that communicates with a process in a malicious way, causing the adversary to hijack the process' control-flow. There are several vulnerabilities that fall into this category: all types of buffer overflows [20] (stack, heap, data, dtors, global offset table, *setjmp* and *longjmp*, double-frees, C++ table of virtual pointers, *etc.*), format string, SQL injection [28] and cross-site scripts (XSS) [30]. Code-reuse attacks [26] are considered a capability of an attacker after leveraging a stack-based buffer overflow and not a vulnerability in itself.

In a general memory corruption attack an adversary provides a victim process with a set of bytes as input, where part of these bytes will overwrite some control information with data of the attacker's choice (usually the address of a malicious instruction). This control information contains data that will eventually be loaded into the EIP register, which contains the address of the next instruction to be executed by the CPU at the architecture level.

For these cases, the fracture in the interpretation of information occurs when user input crosses boundaries of abstractions. User input is able to influence the OS, which manages the process address space and the control memory region being abused. User input also influences the architecture layer as it is directly written into the EIP register. For buffer overflows on the heap, data, and *dtors* areas, an attacker overwrites a data structure holding a function pointer with a malicious address. The effect is the same in all cases: the function will be eventually called, and its address will be loaded into the EIP register.

In a SQL injection [28] user input is directly combined with a SQL command written by an application developer, and this allows an attacker to break out of the data context when she supplies input as a combination of data, control characters and her own code. This malicious combination causes a misinterpretation of data input as it is provided by the web scripting engine. The scripting engine, which processes user input, misinterprets it as data that should be concatenated with a legitimate command created by the application developer.

Table 1. Taxonomy categories.

Category	Description	Examples
Control-flow hijacking	Vulnerabilities where information flows from the input to a process into the control flow of the process causing its execution to be hijacked	Buffer overflows, memory corruption, SQL injection, cross-site scripts
Process confusion	Vulnerabilities where information flows from the security metadata of one object into a security decision about another	TOCTTOU, confused deputy, cross-site request forgery (CSRF)
Side-channels	Vulnerabilities where information flows from physical or side-effects of the operation or communication channels of the system into an illegitimate authentication decision or information disclosure.	Physical: timing/power and electromagnetic attacks. Communications/operation: man-in-the-middle, replay, /proc filesystem attacks
Exhaustion	Vulnerabilities where a significant amount of information flows into a process causing unavailability (exhaustion of resources) or an illegitimate authentication decision (exhaustion of input space)	Resources: DoS, TCP SYN flood, ICMP flood. Input space: password cracking and dictionary attacks
Adversarial accessibility	Vulnerabilities where information is allowed to flow to the attacker's process causing a breach of confidentiality, illegitimate authentication or interference with system functionality	Assignment of weak permissions to system objects, access control errors, and non-control-data attacks [14]

The SQL query interpreter then parses the input provided by the scripting engine as SQL code that should be parsed and executed. The misinterpretation between the web scripting engine and the SQL query interpreter causes the vulnerability.

2.2 Process confusion

This type of vulnerability allows an attacker to confuse a process at a higher layer of abstraction where this process is usually acting as a deputy, performing some task on behalf of another lower privileged process. A fracture in the interpretation of information allows the security metadata of one object to be transferred into a security decision about another object. A classic example is TOCTTOU, one of the oldest and most well-studied types of vulnerability [23]. It occurs when privileged processes are provided with some mechanism to check whether a lower-privileged process should be allowed to access an object before the privileged process does so on the lower-privileged process' behalf. If the object or its attribute can change either between this check and the actual access that the privileged process makes, attackers can exploit this fact to cause privileged processes to make accesses on their behalf that subvert security. The classic example of TOCTTOU is the sequence of system calls `access()` followed by `open()`:

```
if (access("/home/bob/symlink",
     R_OK | W_OK) != -1)
{
     // Symbolic link can change here
     f = fopen("/home/bob/symlink", "rw");
     ...
}
```

What makes this a vulnerability is the fact that the invoker of the privileged process can cause a race condition where something about the filesystem changes in between the call to `access()` and the call to `open()`. For example, the file `/home/bob/symlink` can be a symbolic link that points to a file the attacker is allowed to access during the `access()` check (*e.g.*, file `/home/bob/bob.txt`) that `bob` can read and write, but at a critical moment is changed to point to a different file that needs elevated privileges for access (*e.g.*, `/etc/shadow`).

Consider that the security checks for `/home/bob/bob.txt` (including `stat()`ing each of the dentry's and checking the inode's access control list) get compressed into a return value for the `access()` system call that is stored in register EAX. This information is interpreted to mean that `bob` is allowed to access the file referred to by `/home/bob/symlink`.

The information crosses the boundary between an OS abstraction (the kernel) and a user-level abstraction into the EAX register, which contains the return value (architecture layer abstraction). Then a control flow transfer conditioned on the EAX register is now transformed into a decision to open the file pointed to by `/home/bob/symlink`. The interpretation of information becomes fractured in this information flow between the return value and the `open()` system call,

which occurs at the architecture layer. To the OS, the value returned in register EAX was a security property of `/home/bob/bob.txt`. At the architectural level the value of the program counter (register EIP), *which contains the exact same information,* is implied to be a security property of `/etc/shadow`. The information is the same, but when viewed from different perspectives for the different layers of abstraction that make up the system the interpretation has been fractured.

TOCTTOU is a much broader class of vulnerabilities and no all cases are related to UNIX filesystem atomicity issues [29].

2.3 Side-Channels

This type of vulnerability allows an attacker to learn sensitive information about a system such as cryptographic keys, sites visited by a user, or even the options selected by the user when interacting with web applications by leveraging physical or side-effects of the system execution or communications.

Examples of such vulnerability are found in systems where the execution of certain branches is dependent on input data, causing the program to take varying amounts of time to execute. Thus, an attacker can gain information about the system by analyzing the execution time of algorithms [12]. Other physical effects of the system can be analyzed, such as hardware electromagnetic radiation, power consumption [27] and sound [34]. An attacker can also exploit weaknesses in the communication channels of a process to breach confidentiality [13,19,33].

As example, first consider a timing attack (*Physical side-channel*) where an adversary attempts to break a cryptosystem by analyzing the time a cryptographic algorithm takes to execute [12]. The cryptographic algorithm itself does not reveal cryptographic keys, but the leaking of timing information is a side-effect of its execution. This information flows from the server machine to the client machine and is interpreted in the client (the attacker's machine) as tokens of meaningful information. The combination of these tokens of information over several queries allows the attacker to succeed by making correlations among the input, the time to receive an answer, and the key value.

Another example is a Man-in-the-middle (MiM) vulnerability (*Communications / Operation*), which is a form of *eavesdropping* where the communication between two parties, Alice and Bob, is monitored by an unauthorized party, Eve. The eavesdropping characteristic of MiM vulnerabilities implies that authentication information is *leaked* through a channel not anticipated by the system designer (usually the network). In the classic example, Alice asks for Bob's public key, which is sent by Bob through the communication channel. Eve is able to eavesdrop the channel and intercepts Bob's response. Eve sends a message to Alice claiming to be Bob and passing Eve's public key. Eve then fabricates a bogus message to Bob claiming to be Alice and encrypts the message with Bob's public key. In this attack information flows from the communication channel between Alice's and Bob's processes into an illegitimate authentication decision established by Eve.

2.4 Exhaustion

This type of vulnerability allows an adversary to compromise the availability or confidentiality of a system by artificially increasing the amount of information the system needs to handle. This augmented information flow can leave the system unable to operate normally (attack on availability) or can allow an attacker to illegitimately authenticate herself into the system (attack on confidentiality). The *Exhaustion* category was subdivided into two subcategories (exhaustion of resources and exhaustion of input space) due to their differences in nature and also because they target different security pillars, respectively availability and confidentiality. They both belong to the same broader category because they leverage an artificial increase in the amount of information flowing into the system.

Exhaustion of resources vulnerabilities allow an attacker to cause a steep consumption of a system's computational resources, such as CPU power, memory, network bandwidth or disk space. A classic example is the standard DoS attack: an attacker saturates a target machine with communication requests so that the machine is left short of resources to serve legitimate requests. The victim server process does not handle the uncommon case (exploited by attackers) of a steep increase in the amount of information it has to handle.

Exhaustion of input space vulnerabilities are leveraged to allow an adversary to illegitimately authenticate herself into the system by exploiting a great portion of a vulnerable process authentication input space. For example, in a password cracking attack an adversary repeatedly attempts password strings in the hope that one of them will allow her to authenticate herself into the system. A system will be vulnerable to this type of attack depending on the strength of the password. A secure system can tolerate a steep increase in authentication information flowing into it (password guesses) without its confidentiality being compromised, or guard itself against an exhaustion attack, by for example, locking the system after a few failed attempts.

2.5 Adversarial Accessibility

These vulnerabilities occur when weaknesses in the system design and implementation allow information to flow to an adversary or her process when it should not, as per the system security policies. A classic example is when weak permissions are assigned to system objects, allowing an adversary access to sensitive information or abstractions. This illegitimate information flow to the attacker can also result in authentication breaches. For instance, a vulnerable access control mechanism that does not perform all necessary checks can allow an attacker to authenticate herself in the system and access its resources.

3 Evaluation

The goal of this study was to evaluate how faithfully the categories reflect real vulnerabilities and to assess the taxonomy's potential for classifying vulnerabilities unambiguously. This analysis leveraged three well-known public

vulnerability databases: SecurityFocus (SF) [4], National Vulnerability Database (NVD) [1], and Open Source Vulnerability Database (OSVDB) [2].

The study employed machine learning to classify a large number of vulnerabilities according to the proposed taxonomy. In this analysis we used the Weka data mining software [32]. The study started with the *manual* classification, according to the proposed taxonomy, of 728 vulnerabilities from SecurityFocus (202 vulnerabilities), NVD (280 vulnerabilities), and OSVDB (246 vulnerabilities) databases. This manual classification was done independently by four of the authors, with an inter-rater agreement of approximately 0.70 (see Table 2). A vulnerability report contains the following attributes (names vary per database): ID, title, description, class, affected software and version, reporter, exploit and solution. For purposes of classification, the most important attributes in a vulnerability report are the title and the description. The *class* attribute was observed to be highly ambiguous; SecurityFocus, for instance, classifies highly distinct vulnerabilities as *Design error*. The manual classification selected vulnerabilities in descending chronological order, starting with the most recent vulnerabilities in the respective databases. As some categories were under-represented in the most recent set of reported vulnerabilities and the goal was to build a large and well-represented training set, the authors manually searched for reports fitting under-represented categories in the past. This process showed that the taxonomy was easily applied, even though some questions were raised about vulnerabilities with poor or ambiguous descriptions. Table 3 shows a summary of the manual classification.

Approximately 12 % of the most recent vulnerability reports contain insufficient or ambiguous information to reason about the corresponding security flaw. For example, the SecurityFocus vulnerability report with BID 55977 only reveals that a certain software is vulnerable. To avoid polluting the training set and confusing the machine learning classifier, all vulnerabilities with insufficient or ambiguous descriptions (87 total) were filtered out of the manually labeled set.

The study proceeded with the automated extraction of all vulnerability reports from NVD, OSVDB and SecurityFocus for the periods of 2013-2012, 2009-2008, and 2004-2003. The goal was to classify vulnerabilities from three distinct periods over the last decade and identify trends and patterns. A total of 70919 vulnerabilities were extracted (37030 from OSVDB, 23155 from NVD and 10506 from Security Focus) forming the testing set to be categorized by the machine learning classifier. We used the Naïve Bayes algorithm as it is popular for text classification.

All the reports collected for the training and testing set were pre-processed by a parser that converted them into the Weka's ARFF format [32]. The parser used the Weka's String to Word vector filter [32], which turned each word in the title or description into an attribute, and checked whether or not it was present. The filter removed stopwords and established a threshold on the number of words kept per machine learning sample.

Table 4 summarizes the results obtained for the automated classification of vulnerabilities for the three databases studied. *Control-flow hijacking*

Table 2. Examples of manually classified vulnerabilities.

Category	Database/ID	Description (Abridged)
Control-flow hijacking	SF 54982	*"glibc is prone to multiple stack-based buffer-overflow vulnerabilities because it fails to perform boundary checks on user-supplied data".*
Process confusion	NVD 2013-2709	*"Cross-site request forgery vulnerability in the FourSquare Checkins plugin allows remote attackers to hijack the authentication of arbitrary users".*
Side-channels	OSVDB 94062	*"RC4 algorithm has a cryptographic flaw .. the first byte output by the PRG ... correlating to bytes of the key ... allows attacker to collect keystreams to facilitate an attack"*
Exhaustion	NVD 1999-1074	*"Webmin does not restrict the number of invalid passwords that are entered for a valid username, ... allow remote attackers to gain privileges via brute force password cracking.*
Adversarial accessibility	NVD 2013-0947	*"EMC RSA Authentication Manager allows local users to discover cleartext operating-system passwords ... by reading a log file or configuration file."*
No information	SF 55977	*"Oracle Outside In Technology is prone to a local security vulnerability. The 'Outside In Filters' sub component is affected. Oracle Outside In Technology is vulnerable."*
Ambiguous	SF 39710	*JBoss is prone to multiple vulnerabilities, including an information-disclosure issue and multiple authentication-bypass issues. An attacker can exploit these issues to bypass certain security restrictions to obtain sensitive information..."*

Table 3. Manual classification of vulnerabilities.

Database	Control-flow hijacking	Process confusion	Side channels	Exhaustion	Adversarial accessibility	No info	Ambiguous
SF (202)	60 (30 %)	32 (16 %)	27 (13 %)	34 (17 %)	18 (9 %)	17 (8 %)	14 (7 %)
NVD (280)	149 (53 %)	8 (3 %)	24 (8 %)	35 (12 %)	30 (11 %)	11 (4 %)	23 (8 %)
OSVD (246)	150 (61 %)	9 (4 %)	26 (10 %)	32 (13 %)	15 (6 %)	8 (3 %)	3 (1 %)
Total (728)	359 (49 %)	49 (7 %)	77 (10 %)	101 (14 %)	63 (9 %)	36 (5 %)	51 (7 %)

vulnerabilities make more than 50 % of all reported vulnerabilities in all databases, followed by *Adversarial accessibility* (19 %), *Exhaustion* (16 %), *Side-channels* (3 %) and *Process confusion* (2 %). This trend was consistent in all databases and did not change much over the last decade.

The standard method of stratified tenfold cross validation [32] was used to predict the success rate of the classifier, which obtained, respectively, success rates of 84.6 %, 73.1 %, and 82 % for the OSVDB, NVD, and SecurityFocus databases. The authors believe that two reasons prevented the classifiers from obtaining higher success rates: (i) the non-negligible number of reports with insufficient information about the vulnerability; approximately 12 % for the most recent vulnerabilities appearing in the training set for all three databases, and (ii) DoS vulnerabilities, which depending on how they are exploited can be classified as *Exhaustion* or *Control-flow hijacking*. For example, an attack that works by sending a very large number of requests to a server, so as it does not have sufficient resources to serve legitimate requests exploits an *Exhaustion* vulnerability. On the other hand, a buffer overflow that crashes the application (still changing the control-flow according to the attacker's choice) is usually named a DoS attack in vulnerability reports, even though the root cause of the vulnerability does not involve exhaustion of resources. Table 5 shows examples of vulnerabilities automatically categorized by the classifier.

3.1 Discussion

Approximately 12 % of all examined reports do not provide sufficient information to understand the corresponding vulnerabilities. These descriptions specify the capabilities of attackers after the vulnerability is exploited, or just mention that an unspecified vulnerability exists.

Also, important information on the process of finding vulnerabilities is usually not provided: reporter contact information, tools used to discover vulnerabilities, whether the vulnerability was discovered through normal software usage or careful inspection, exploit examples and steps to reproduce the vulnerability. Certain reports provide URLs for exploits or steps to reproduce the flaw, but many of these links are invalid as if this information were ephemeral. This information should be permanently recorded; it is invaluable to educate developers during the software development cycle and help the security community build a body of knowledge about the nature of vulnerabilities.

The lack of this important information in vulnerability reports shows that the roles played by reporters and developers are undermined. Reports discussing strategies for finding vulnerabilities could help developers designing more secure software. Further, it would be invaluable to the security community and other developers information on how the vulnerability was addressed. For example, was the vulnerability caused by a weakness on a particular API ? Did the developer use a particular tool or strategy to address the vulnerability?

Table 4. Automated classification of vulnerabilities.

Period	Total	Control-flow hijacking	Process confusion	Side-channels	Exhaustion	Adversarial accessibility
OSVDB						
2013-12	14270	9261 (64.8 %)	555 (3.8 %)	440 (3 %)	1521 (10.6 %)	2493 (17.4 %)
2009-08	16945	10770 (63.5 %)	66 (0.4 %)	126 (0.7 %)	3099 (18.2 %)	2884 (17 %)
2004-03	5815	2990 (51.4 %)	0	21 (0.3 %)	1318 (22.6 %)	1486 (25.5 %)
All	37030	23021 (62.1 %)	621 (1.6 %)	587 (1.5 %)	5938 (16 %)	6863 (18.5 %)
NVD						
2013-12	7822	4062 (51.9 %)	239 (3 %)	321 (4.1 %)	1141 (14.5 %)	2059 (26.3 %)
2009-08	11361	7021 (61.7 %)	207 (1.8 %)	310 (2.7 %)	1388 (12.2 %)	2435 (21.4 %)
2004-03	3972	1958 (49.2 %)	57 (1.4 %)	132 (3.3 %)	690 (17.3 %)	1135 (28.5 %)
All	23155	13041 (56.3.1 %)	503 (2.1 %)	763 (3.2 %)	3219 (13.9 %)	5629 (24.3 %)
SecurityFocus						
2013-12	2071	1057 (51 %)	122 (5.8 %)	60 (2.8 %)	335 (16.1 %)	497 (23.9 %)
2009-08	5788	4216 (72.8 %)	172 (2.9 %)	168 (2.9 %)	661 (11.4 %)	571 (9.8 %)
2004-03	2647	710 (26.8 %)	1264 (47.7 %)	512 (19.3 %)	1264 (47.7 %)	139 (5.2 %)
All	10506	5983 (56.9 %)	316 (3 %)	750 (7.1 %)	2260 (21.5 %)	1207 (11.4 %)
All databases consolidated						
2013-12	24163	14380 (59.5 %)	916 (3.7 %)	820 (3.3 %)	2997 (12.4 %)	5049 (20.8 %)
2009-08	34094	22007 (64.5 %)	445 (1.3 %)	604 (1.7 %)	5148 (15 %)	5890 (17.2 %)
2004-03	12434	5658 (45.5 %)	1321 (10.6 %)	665 (5.3 %)	3272 (26.3 %)	2790 (22.4 %)
All	70691	42045 (59.4 %)	1440 (2 %)	2100 (2.9 %)	11417 (16.1 %)	13699 (19.3 %)

A lack of standardization among vulnerability reports across databases was also observed. This makes it very difficult to understand actual trends and statistics about vulnerabilities; they are viewed as one of a kind and not addressed together according to their similarities. Finally, there is no guarantee that a vulnerability is reported in a public database only after the vendor had been informed about the issue. A responsible reporter should always report the vulnerability first with the vendor or developer and allow them a reasonable amount of time (*e.g.*, 30 days) to address the issue before making it public in a database.

4 Related Work

The first efforts towards understanding software vulnerabilities happened in the 70 s through the RISOS Project [7] and the Protection Analysis study [18]. Landwehr *et al.* [21] proposed a taxonomy based on three dimensions: genesis, time, and location, and classified vulnerabilities as either intentional (malicious and non-malicious) or inadvertent. Aslam [8] introduced a taxonomy targeting the organization of vulnerabilities into a database and also the development of static-analysis tools. Bishop and Bailey [9] analyzed these vulnerability taxonomies and concluded that they were imperfect because, depending on the layer of abstraction that a vulnerability was being considered in, it could be classified in multiple ways.

Table 5. Examples of vulnerabilities automatically categorized by the classifier.

Category	Database/ID	Description (Abridged)
Control-flow hijacking	NVD 2003-0375	*"XSS vulnerability in member.php of XMBforum XMB allows remote attackers to insert arbitrary HTML and web script via the "member" parameter."*
Process confusion	OSVDB 94899	*"DirectAdmin Backup System contains a flaw as an unspecified email account function creates temporary files insecurely. It is possible for attacker to use a symlink attack against an unspecified file to gain elevated privileges".*
Side-channels	OSVDB 95626	*"WhatsApp Messenger contains a flaw triggered when attacker intercepts a payment request via a MiM attack ... allow the attacker to redirect user to arbitrary web page"*
Exhaustion	SF 58500	*"IBM Integrator is prone to a DoS vulnerability. Remote attackers can exploit this issue to cause an application to consume excessive amounts of memory and CPU time, resulting in a DoS condition"*
Adversarial accessibility	NVD 2013-3431	*Cisco Video Surveillance Manager does not require authentication for access to VSMC monitoring pages, allows remote attackers to obtain sensitive configuration information.*

Lindqvist and Jonsson [22] presented a classification of vulnerabilities with respect to the intrusion techniques and results. The taxonomy on intrusion techniques has three global categories (Bypassing Intended Controls and Active and Passive Misuse of Resources), which are subdivided into nine subcategories. The taxonomy on intrusion results has three broader categories (Exposure, Denial of Service and Erroneous Output), which are subdivided into two levels of subcategories.

More recently the Common Weakness Enumeration (CWE) [6] was introduced as a dictionary of weaknesses maintained by the MITRE Corporation to facilitate the use of tools that can address vulnerabilities in software. The Open Web Application Security Project (OWASP) was also created to raise awareness about application security by identifying some of the most critical risks facing organizations. Even though these projects do not define themselves as taxonomies, their classification is ambiguous. For example, CWE-119 and

CWE-120 are two separate weaknesses that address buffer overflows. Also, OWASP classifies *injection* and XSS as different categories, even though XSS concerns malicious code being injected into a web server.

There are also discussions about the theoretical and computational science of exploit techniques and proposals to do explicit parsing and normalization of inputs [11,16,24,25]. Bratus *et al.* [11] discuss "weird machines" and the view that the theoretical language aspects of computer science lie at the heart of practical computer security problems, especially exploitable vulnerabilities. Samuel and Erlingsson [25] propose input normalization via parsing as an effective way to prevent vulnerabilities that allow attackers to break out of data contexts. Crandall and Oliveira [16] discussed in a position paper the information-flow nature of software vulnerabilities.

In this work vulnerabilities are viewed as fractures in the interpretation of information as it flows in the system. It is not attempted to pinpoint a location for a vulnerability because they can manifest in several locations or semantic boundaries. Further, the primary goal of our taxonomy is to address ambiguity, which makes it difficult to reason about vulnerabilities effectively.

5 Conclusions

This paper presented a new vulnerability taxonomy that views vulnerabilities as fractures in the interpretation of information as it flows in the system. Notorious vulnerabilities are discussed in terms of the taxonomy's categories. A machine learning study evaluating the taxonomy is presented. Almost 71000 vulnerabilities were automated classified with an average success rate of 80 %. The results showed the taxonomy's potential for unambiguous understanding of vulnerabilities. Lessons learned were discussed: (i) control-flow hijacking vulnerabilities represent more than 50 % of all vulnerabilities reported, a trend that was not changed over the last decade, (ii) approximately 12 % of recent vulnerabilities reports have insufficient information about the security flaw, (iii) the lack of standards in reporting makes it difficult to address vulnerabilities scientifically. This work will hopefully shed light on how the security community should approach vulnerabilities and how to best develop automatic diagnostic tools that find vulnerabilities automatically across layers of abstraction.

References

1. National Vulnerability Database. http://nvd.nist.gov/home.cfm
2. Open Source Vulnerability Database. http://www.osvdb.org/
3. Security Focus Vulnerability Notes, bugtraq id 66483. http://www.securityfocus.com/bid/66483
4. SecurityFocus. http://www.securityfocus.com/
5. Symantec - Internet Security Threat Report. http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf
6. The Common Weakness Enumeration (CWE). http://nvd.nist.gov/cwe.cfm

7. Abbot, R.P., Chin, J.S., Donnelley, J.E., Konigsford, W.L., Webb, D.A.: Security Analysis and Enhancements of Computer Operating Systems. NBSIR 76–1041, Institute for Computer Sciences and Technology, National Bureau of Standards (1976)
8. Aslam, T.: A Taxonomy of Security Faults in the UNIX Operating System (1995)
9. Bishop, M., Bailey, D.: A Critical Analysis of Vulnerability Taxonomies. Technical Report CSE-96-11, University of California at Davis (1996)
10. Borisov, N., Johnson, R., Sastry, N., Wagner, D.: Fixing races for fun and profit: how to abuse atime. In Proceedings of the 14th conference on USENIX Security Symposium, SSYM 2005, vol. 14, Berkeley, CA, USA. USENIX Association, p. 20 (2005)
11. Bratus, S., Locasto, M.E., Patterson, M.L., Sassaman, L., Shubina, A.: Exploit programming.: From buffer overflows to "Weird Machines" and theory of computation. USENIX: login, December 2011
12. Brumley, D., Boneh, D.: Remote timing attacks are practical. USENIX Security (2003)
13. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-channel leaks in web applications: A reality today, a challenge tomorrow. In: IEEE Symposium on Security and Privacy (2010)
14. Chen, S., Xu, J., Sezer, E.: Non-control-hijacking attacks are realistic threats. In: USENIX Security (2005)
15. Cohen, F.: Computer viruses: Theory and experiments. In: 7th DoD/NBS Computer Security Conference Proceedings, pp. 240–263, September 1984
16. Crandall, J., Oliveira, D.: Holographic vulnerability studies: Vulnerabilities as fractures in terpretation as information flows across abstraction boundaries. In: New Security Paradigms Workshop (NSPW) (2012)
17. Dean, D., Hu, A.J.: Fixing races for fun, profit: How to use access(2). In: Proceedings of the 13th Conference on USENIX Security Symposium, Berkeley, CA, USA, vol. 13, p. 14. USENIX Association (2004)
18. Bisbey II, R., Hollingsworth, D.: Protection Analysis Project Final Report. ISI/RR-78-13, DTIC AD A056816, USC/Information Sciences Institute (1978)
19. Jana, S., Shmatikov, V.: Memento: Learning secrets from process footprints. In: IEEE Symposium on Security and Privacy (2012)
20. Kyung-Suk, L., Chapin, S.J.: Buffer overflow, format string overflow vulnerabilities. Softw. Pract. Experience **33**(5), 423–460 (2002)
21. Landwehr, C.E., Bull, A.R., McDermott, J.P., Choi, W.S.: A taxonomy of computer program security flaws. ACM Comput. Surv. **26**(3), 211–254 (1994)
22. Lindqvist, U., Jonsson, E.: How to systematically classify computer security intrusions. In: IEEE Symposium on Security and Privacy (1997)
23. McPhee, W.S.: Operating system integrity in OS/VS2. IBM Syst. J. **13**(3), 230–252 (1974)
24. Pieters, W., Consoli, L.: Vulnerabilities and responsibilities: dealing with monsters in computer security. J. Inf. Commun. Ethics Soc. **7**(4), 243–257 (2009)
25. Samuel, M., Erlingsson, U.: Let's parse to prevent pwnage (invited position paper). In: Proceedings of the 5th USENIX Conference on Large-Scale Exploits and Emergent Threats, LEET 2012, Berkeley, CA, USA, p. 3. USENIX Association (2012)
26. Shacham, H.: The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In: ACM CCS, pp. 552–561 (2007)
27. Spadavecchia, L.: A network-based asynchronous architecture for cryptographic devices. Edinburgh Research Archive (2005)

28. Su, Z., Wassermann, G.: The essence of command injection attacks in web applications. In: Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006, pp. 372–382. ACM, New York (2006)
29. Wang, R., Chen, S., Wang, X., Qadeer, S.: How to shop for free online - security analysis of cashier-as-a-service based web stores. In: Proceedings of the 2011 IEEE Symposium on Security and Privacy, SP 2011, pp. 465–480. IEEE Computer Society (2011)
30. Wassermann, G., Su, Z.: Static detection of cross-site scripting vulnerabilities. In: 30th International Conference on Software Engineering, ICSE 2008. ACM, New York (2008)
31. Weber, S., Karger, P.A., Paradkar, A.: A software flaw taxonomy: aiming tools at security. In: Software Engineering for Secure Systems (SESS) (2005)
32. Witten, I.W., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
33. Zhang, K., Wang, X.: Peeping tom in the neighborhood: keystroke eavesdropping on multi-user systems. USENIX Security (2009)
34. Zhuang, L., Zhou, F., Tygar, J.D.: Keyboard acoustic emanations revisited. In: ACM Conference on Computer and Communications Security (CCS) (2005)
35. Security Focus Vulnerability Notes. bid == Bugtraq ID. http://www.securityfocus.com

XSS PEEKER: Dissecting the XSS Exploitation Techniques and Fuzzing Mechanisms of Blackbox Web Application Scanners

Enrico Bazzoli[1], Claudio Criscione[2], Federico Maggi[1(✉)], and Stefano Zanero[1]

[1] Politecnico di Milano, Milano, Italy
federico.maggi@polimi.it
[2] Google Zürich, Zürich, Switzerland

Abstract. Black-box vulnerability scanners can miss a non-negligible portion of vulnerabilities. This is true even for cross-site scripting (XSS) vulnerabilities, which are relatively simple to spot. In this paper, we focus on this vulnerability class, and systematically explore 6 black-box scanners to uncover how they detect XSS vulnerabilities, and obtain useful insights to understand their limitations and design better detection methods. A novelty of our workflow is the retrofitting of the testbed so as to accommodate payloads that triggered no vulnerabilities in the initial set. This has the benefit of creating a systematic process to increase the number of test cases, which was not considered by previous testbed-driven approaches.

1 Introduction

Web application vulnerabilities are one of the most important and popular security issues, constantly making it to the top list of disclosed vulnerabilities [13,18]. Cross-site scripting (also known as XSS) is a prevalent class of web application vulnerabilities [10]. In June 2013, XSS was reported as the most prevalent class of input validation vulnerabilities by far [16].

Black-box vulnerability scanners are widely used in the industry but, unfortunately, they can miss a non-negligible portion of vulnerabilities [2,14,17] or report non-existing, non-exploitable or uninteresting vulnerabilities [17]. Although XSS vulnerabilities are relatively easy to discover, previous work showed that black-box scanners exhibit shortcomings even in the case of XSS flaws.

To our knowledge there is no detailed study of black-box web vulnerability scanners that focused specifically on XSS vulnerabilities and their detection approaches. Previous work and commercial benchmarks included XSS bugs as part of the set of flaws in testbed web applications, but not as the main focus. Also, previous work measured the detection rate and precision of the scanners mainly with the goal of benchmarking their relative performance. Although these

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 243–258, 2016.
DOI: 10.1007/978-3-319-33630-5_17

indicators are important, we believe that providing precise insights on the structure, generality, fuzzing mechanism and overall quality of the XSS detection approaches could help web application developers to design better escaping and validation routines, and scanner vendors to *understand* the reasons behind scanner weaknesses.

These goals require a new perspective. A naïve approach would be to launch the scanners against large set of web applications, with difficult-to-find entry points, complex session mechanisms, etc. Normally, this kind of testbeds are adopted by vendors to demonstrate their product's sophistication. However, since our goal is *not* to challenge a scanner, but to analyze its payloads, exploitation approaches and fuzzing algorithms, we need a comprehensive set of targeted test cases that the scanner can easily find and exercise, such that the number of extracted payloads is maximized. From here the name of our tool, "XSS Peeker," that implements our workflow.

XSS Peeker works at the network level and decodes the HTTP layer to find and extract the XSS payloads. In addition, XSS Peeker streamlines tedious tasks such as finding groups of related payloads, which we call *templates*, making automatic analysis and result visualization feasible even in the case of large amounts of payloads. Our research is not limited to individual scanners: we observed how cross-scanner analysis yields very interesting results.

Part of our contributions is a testbed web application, *Firing Range* (http://public-firing-range.appspot.com) It is designed such that it is easy for the scanner to find the vulnerable entry points. Moreover, it is very easy to add new vulnerable entry points. Indeed, as part of our workflow, whenever XSS Peeker encounters a payload that does not trigger any vulnerability, it displays an informative alert to the developer who can quickly prepare a test case that would satisfy specifically that payload. We applied this process iteratively, running new scans and collecting new payloads. Last, a difference of our testbed application with respect to the state of the art is the inclusion of specific test cases for DOM XSS vulnerabilities, one of the most recently discovered and emerging sub-classes of XSS flaws [5,7].

Overall, our methodology and results are useful to answer questions such as: which payloads are used by security scanners to trigger XSS vulnerabilities? Can we rank them and identify a set of "better" payloads? Is a scanner exercising all the entry points with the expected payloads? A direct application of our results toward improving the quality of a scanner is, for example, to minimize the size and number of characters used in the payloads, while keeping the detection rate constant. Payloads of short size and with a smaller character set have in general higher chances of bypassing filters, and thus higher possibilities of uncovering unknown vulnerabilities. Using better payloads also means generating fewer requests and thus completing a test in a shorter time (i.e., better scanning efficiency). Clearly, these are just a couple of examples, which we formalize through a set of criteria later in the paper. This type of analysis should be focused on a scanner testing phase in isolation, which is best achieved with a fully synthetic test bed: a realistic test application would not improve the

quality of our analysis in any of our measures and would introduce yet another experimental parameter to factor in all our considerations.

In summary, we make the following contributions:

- A publicly available testbed web application that exposes a wide range of non-trivial XSS vulnerabilities, augmented with the new cases progressively discovered while running the scanners.
- A methodology to analyze how black-box web application scanners work by (1) extracting the payloads from the HTTP requests, (2) clustering them to scale down the evaluation challenge and keep it feasible, and (3) evaluating each cluster in terms of use of evasion, compactness, and other quality indicators.
- A publicly available prototype of XSS Peeker (https://bitbucket.org/necst/xss-peeker)
- A detailed evaluation of 6 scanners: Acunetix 8.0, NetSparker 3.0.15.0, N-Stalker 10.13.11.28, NTOSpider 6.0.729, Skipfish 2.10b and w3af 1.2.

2 Background

Before describing our approach in detail, we introduce the essential background concepts and terminology on web application XSS vulnerabilities and black-box scanners.

XSS *attacks* consist in the execution of attacker-controlled code (e.g., JavaScript) in the context of a vulnerable web application. In this paper, we refer to the portion of malicious code as *payload*. Without aiming for a complete taxonomy, XSS vulnerabilities and attacks can be divided in *stored* and *reflected*. In reflected attacks the victim is tricked (e.g., through links or short URLs [8] embedded in e-mail or instant messages) into sending a specially crafted request—which embeds the actual payload, which is bounced back to the client immediately. In stored XSS attacks the moment the payload is injected is decoupled from the moment that it is effectively displayed and executed by the victim, as the attacker's code achieves some form of persistence.

DOM-based attacks [6] can be seen as an orthogonal category. They rely on the insecure handling of untrusted data through JavaScript (e.g., `document.write('<script ...>')`) rather than static inclusion of payload (e.g., `<script ...>`) in the rendered page.

Black-box web vulnerability scanners leverage a database of known exploits, including XSS payloads, used to trigger and detect potential vulnerabilities. They start by *crawling* the target web application to enumerate all reachable *entry points* (e.g., links, input fields, cookies), then they generate (mutations of) input strings based on their database, inject the resulting payload in the entry points and finally analyze the HTTP responses using an *oracle* to validate the presence of vulnerabilities (e.g., by looking for the injected payload in the output).

3 Firing Range: Test Case Generation

The implementation of the testbed web application is a key point. Given our goals and needs, the requirements of such a testbed are: (i) to have clearly defined vulnerabilities and entry points, (ii) to be easily customizable, (iii) to contain the main types of XSS vulnerabilities. Large, full-fledged testbed web applications have been implemented in previous works [1–3] and are constantly made available to the public, but they do not entirely meet our requirements and approach. Although requirement (iii) is easy to ensure by modifying existing testbed applications, ensuring (i) and (ii) implies a complete redesign and re-engineering. In fact, existing testbed applications are not focused on *extracting* as many payloads as possible from the scanner. Contrarily, they are focused on challenging the scanner's capabilities of discovering hard-to-find vulnerable entry points. Given these premises, we decided that it was easier to implement our own testbed and release it to the community.

3.1 Design Challenges

Meeting the above requirements is tricky. On one hand, as pointed out in [2], the complexity of some applications can hinder the coverage of the scanner. On the other hand, there is no guarantee of comprehensiveness of the testbed.

We decided to address these shortcomings explicitly while designing *Firing Range*. Our testbed exposes all vulnerabilities through HTML anchors, and vulnerable parameters are provided with sample input to improve discoverability. This approach allows to run testing sessions by providing the full list of vulnerable URLs, thus removing any crawling-related failure entirely. Furthermore, each vulnerable page is served as a standalone component with no external resources such as images or scripts, as to create a minimal, clean and efficient test case. The result is that the scanner focuses on the juicy part: the exploit generation and fuzzing.

3.2 Implementation Challenges

The main implementation challenge when creating a testbed is of course deciding which tests to include. We wanted our initial set to cover as many cases as possible, but there was no such list readily available in previous literature.

Since we basically wanted to test the detection of XSS on an HTML page, we observed that the HTML parsers in modern browsers have a finite set of states, which make a good starting point to create test cases. Thus, we created one test per HTML parser state. To this end we analyzed the different contexts identified by the contextual auto-escaper described in [12]: simply reversing the perspective of a parser, tasked with applying proper escaping to malicious payloads, provided us with clear samples of the different contexts. We generated test cases covering each of them with the goal of (1) producing a "vulnerable baseline" of the different states of an HTML parser and (2) inducing the scanners to inject as many payloads as possible.

HTML contexts are however not enough to generate test cases for DOM XSSs, which exploits interactions between the DOM generated by parsing the original HTML and JavaScript code. For DOM XSS, we started from the XSS Wiki[1], and other openly available collections of sample vulnerabilities, and generated a list of valid DOM sinks and sources—which, notably, include sources other than URLs such as cookies and browser storage. Each one of our DOM tests couples one of these sinks and sources. In the following example, the source is `location.hash` and the sink is `innerHTML` of a `<div>` node:

DOM XSS from location.hash to innerHTML.

```
<body>
  <script>
    var payload = window.location.hash.substr(1);
    var div = document.createElement('div');
    div.id ='divEl';
    document.documentElement.appendChild(div);

    var divEl = document.getElementById('divEl');
    divEl.innerHTML = payload;
  </script>
</body>
```

We then varied the sources and sinks, obtaining test cases like the following ones:

DOM XSS from `documentURI` to `document.write()`.

```
<body>
  <script>
    var payload = document.documentURI;
    document.write(payload);
  </script>
</body>
```

DOM XSS from `window.name` to `eval()`.

```
<body>
  <script>
    var payload = window.name;
    eval(payload);
  </script>
</body>
```

We manually verified all the initial tests as exploitable with an appropriate payload or attack technique.

3.3 Iteratively Discovered Test Cases

During our experimental evaluation XSS Peeker discovered payloads that were not exercising any test case (i.e., vulnerability). Instead of limiting our analysis to report this gap, our approach is to iteratively construct new test cases and progressively re-run all the scanners. In other words, our testbed application is dynamic by design. The details of this iterative procedure are described in Sect. 4.4, whereas the numbers of test cases that we had to add in order to accommodate an existing exploitation payload are reported in Sect. 5.4.

This iterative process produced 42 new test cases that were not identified by our initial seeding. In other word, we use the testbed web application as an

[1] https://code.google.com/p/domxsswiki/wiki/Introduction.

oracle with respect to the scanner, and we use the scanner as an oracle with respect to the web application. As a result, this dual approach greatly improved the testbed and provided new insights on the internals of each scanner.

When considering XSS Peeker's analysis perspective, there is no functional difference between stored and reflected XSS. The difference is whether the echoed payload is stored or not. However, from the point of view of the payload analysis, which is our core focus, reflected or stored XSSs are equivalent. Therefore, for ease of development and of experimental repeatability, our testbed web application only contains reflected vulnerabilities.

Our publicly available testbed includes full details of the vulnerabilities. For the sake of brevity, we refer the reader directly to http://public-firing-range.appspot.com for a complete list of live test cases and commented source code.

4 XSS Peeker: Analysis Workflow

XSS Peeker automates the extraction and analysis of XSS payloads by following an iterative approach, divided in four phases (the first three completely automated, the fourth partially relying on manual inputs).

4.1 Phase 1 (Payload Extraction)

The high-level goal of this phase is to obtain, for each scanner, the entire set of XSS payloads used by each scanner for each entry point in the testbed application. To this end, this phase first captures (using `libpcap`) the traffic generated by the scanners during the test and performs TCP stream reassembly to the full HTTP requests.

The first challenge is to automatically separate the HTTP requests used for the actual injection (i.e., containing one or more payload) from the requests used for crawling or other ancillary functionalities, which are not interesting. The ample amount of payloads generated makes manual approaches unfeasible. Therefore, we rely on two heuristics:

Signature Payloads: Most scanners use *signature payloads* (i.e., payloads that contain strings that are uniquely used by that scanner). Therefore, we derived the signature payloads for each scanner and compiled a whitelist that allows this heuristic to discard the uninteresting requests.

Attack Characters: Since we know the testbed application, we guarantee that there is no legitimate request that can possibly contain certain characters in the header or body parameters. These characters include, for example, `<`, `>`, `'`, `"`, and their corresponding URL-percent encodings. Such characters should not be present in a crawling request by construction, and since they are often required to exploit XSS vulnerabilities, we have empirically observed them as linked to vulnerability triggering.

To complement the previous heuristics and maximize the number of identified payloads, we perform pairwise comparisons between requests issued by each couple of scanners. For each couple, we extract the body and URL of the two requests and check if they have the same path and the same query parameters. If so, we compare the values of each query parameter. By construction, Firing Range provides only a single value for each parameter, thus any mismatch has to be originated by the scanner fuzzing routine. Once a pair of requests is flagged as a mismatch, we performed manual inspection to isolate the payload. The number of such cases is rare enough to make this a manageable process. We iteratively applied and improved these heuristics until this cross-scanner analysis generated empty output, and we could confirm through manual inspection that no more test requests were missed (i.e., all payloads considered).

We can focus just on query parameters because of the design of our testbed, which provides injection points exclusively on query parameters. Even if almost all scanners also test path injection, we chose not to analyze them. Indeed, manual analysis confirmed that scanners used the very same set of payloads observed during parameter injection.

4.2 Phase 2 (Payload Templating)

Given the large number of payloads generated by each scanner, manually analyzing and evaluating each of them separately is practically unfeasible. A closer inspection of the payloads, however, revealed self-evident clusters of similar payloads. For example, the following payloads: `<ScRiPt >prompt(905188)</ScRiPt>` and `<ScRiPt>prompt(900741)</ScRiPt>` differ only for the parameter value. To cluster similar payloads, inspired by the approach presented in [11], we developed a recursive algorithm for string templating. Without going into the details, the approach presented in [11] is to start from a set of template elements that produce fully random or dictionary based sequences of symbols. Using a large corpus of spam emails, the approach is to derive the full spam email template by generalizing the template elements. Emails, however, are much larger, structured, and richer of contextual symbols than our exploitation payloads. Therefore, the approach described in [11] cannot be applied as is. Essentially, we cannot easily define the concept of "template elements" in such a small space. Therefore, as part of our contributions, we create a new approach that is well suited for short strings. In a nutshell, rather than following a top-down approach that starts from template elements, our idea is to elicit the template following a bottom-up approach, starting from the strings that it supposedly generated.

More formally, a *template*, in our definition, is a string composed by *lexical tokens* (e.g., a parenthesis, a function name, an angular bracket), that are common to all the payloads that generated it, and *variable parts*, which we represent with placeholders. The `NUM` placeholders replace strings that contains only digits, whereas the `STR` placeholders replace strings that contains alphanumeric characters. For instance, the template for the above example is `<ScRiPt >prompt(90_NUM_)</ScRiPt>`

To generate the templates we leveraged the Levenshtein (or edit) distance (i.e., the minimum number of single-character insertions, deletions, or substitutions required to transform string A to string B).

At each recursion, our algorithm receives as an input a list of strings and performs a pairwise comparison (without repetition) between elements of the input list. If the Levenshtein distance between each two compared strings is lower than a fixed threshold, we extract the matching blocks between the two strings (i.e., sequences of characters common to both strings). If the length of all matching blocks is higher than a given threshold, the matches are accepted. Non-matching blocks are then substituted with the corresponding placeholders. The output of each recursion is a list of generated templates. All payloads discarded by the Levenshtein or matching-block thresholding are appended to the list of output templates, to avoid discarding "rarer" payloads (i.e., outliers) and losing useful samples. The thresholds (maximum Levenshtein distance and minimum matching block length) are decremented at each cycle by an oblivion factor, making the algorithm increasingly restrictive. We selected the parameters, including the oblivion factor, through empirical experimentation, by minimizing the number of templates missed. This automatic selection yielded the following values: 20, 0.9 (default case); 20, 0.9 (Acunetix), 15, 0.5 (NetSparker); 15, 0.8 (NTOSpider); 15, 0.9 (Skipfish); 15, 0.5 (W3af). The algorithm stops when a recursion does not yield any new templates.

4.3 Phase 3 (Template Evaluation)

We want to assess the quality of payloads in terms of *filter-evasion* capabilities and amount of *mutations* used by the scanner. Given our observations above, we apply such evaluation to templates, as opposed to each single payload.

More precisely, the quality of a template is expressed by the following template metrics, which we aggregate as defined in Sect. 5.3. Note that the *rationale* behind each metric is explained on payloads, whereas the metric itself is *calculated* on the templates.

M1 (Length), int: The longer a payload is, the easier to spot and filter (even by accident). Thus, we calculate the length of each payload template to quantify the level of evasion capability.

M2 (Number of distinct characters), int: The presence of particular characters in a ayload could hit server-side filters, or trigger logging. The presence of a character instead of another could reveal an attempt to mutate the string (e.g., fuzzing). A symbol can have different meanings depending on the actual context. From this rationale we obtain that a payload with a small set of characters is "better" than one leveraging rare characters. We calculate this metric on the variable part of each template (i.e., excluding the `STR` ad `NUM` tokens).

M3 (Custom callbacks) bool: Rather than using standard JavaScript functions like `alert`, a scanner can use custom JavaScript function callbacks to bypass simple filters. We interpret this as an evasion attempt. If a template

contains a function outside the set of built-in JavaScript functions, we set this metric to true.

M4 (Multiple encodings), bool: Encoding a payload may let it pass unnoticed by some web applications' filters. However, some applications do not accept certain encodings, resulting in the application not executing the payload. A payload that uses multiple encodings is also more general because, in principle, it triggers more state changes in the web application. We set this metric to true if the template contains symbols encoded with a charset other than UTF-8 and URL-percent, thus quantifying the level of evasion.

M5 (Number of known filter-evasion techniques), int: With this metric we quantify the amount of known techniques to avoid filters in web applications. For each template we calculate how many known techniques are used by matching against the OWASP list[2].

Although other metrics could be designed, we believe that these metrics are the bare minimum to characterize a scanner's capabilities and understand more deeply the quality of the payloads that it produces and process.

4.4 Phase 4 (Retrofitting Negative Payloads)

At the end of a scan, each scanner produces a report of the detected vulnerabilities. We use a report-parsing module that we developed (and released) for each scanner, and correlate the results with the payloads extracted. In this way we identify payloads that triggered vulnerabilities, which we call *positive payloads* and those that did not, called *negative payloads*.

We manually verified each negative payload to ensure that it was not our report-parsing module failing to correlate. We found that there are at least four reasons for which a negative payload occur:

- The payload was malformed (e.g., wrong or missing characters, incorrect structure) and it was not executed. This is a functional bug in the scanner.
- The payload was designed for a different context than the one it was mistakenly injected in.
- The scanner used what appears to be the "right" payload for the test case, but the detection engine somehow failed to detect the exploit.
- The payload was redundant (i.e., the scanner already discovered a vulnerability) in the same location thanks to another payload, and thus will not report it again.

Since one of our goals was to create a testbed application as complete as possible, we wanted to ensure that all negative payloads had a matching test case in our application. With manual analysis, we proceeded to discard malformed and redundant payloads from the list of negative payloads. For each remaining negative payloads we produced a specific vulnerable test case.

[2] https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.

To avoid introducing a bias in the results, we crafted each new test case to be triggered exclusively by the payload type for which it has been created, whereas the other payloads of the same scanner are rejected, filtered, or escaped. Of course, nothing would prevent other scanners from detecting the case with a different payload and that was indeed the intended and expected behavior.

5 Experimental Results

We tested 4 commercial scanners (in random order, Acunetix 8.0, NetSparker 3.0.15.0, N-Stalker 10.13.11.28, NTOSpider 6.0.729), for which we obtained dedicated licenses with the support of the vendors, and 2 open-source scanners (in random order, Skipfish 2.10b, and w3af 1.2). Because performing a comparative analysis is *not* the point of this paper, the scanners are weakly anonymized and they appear as `Scanner1`, `Scanner2`, etc., in our results.

We installed each scanner on a dedicated virtual machine (VM) to guarantee reproducibility and isolation (i.e., Debian 6.0.7 for Skipfish and W3af, and Windows 7 Professional for Acunetix, NetSparker, N-Stalker and NTOSpider). We used Wireshark to capture the traffic. When possible, we configured each scanner to only look for XSS vulnerabilities, and to minimize the impact of other variables, we left the configuration to its default values and kept it unchanged throughout all the tests.

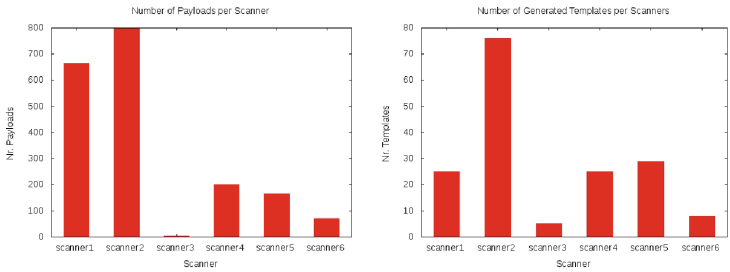
We tested Firing Range several times with each scanner. No scanner reported false positives, which is an expected result since we did not design any test cases to trick them like Bau et al. [1] did in their testbed application.

Of course, simply running scanners against a test application and analyzing their reports is not enough to evaluate their payloads. As Doupé et al. [2] did in their study, we wanted to understand the behavior of a scanner in action to be able to explain their results. Our approach, however, differs noticeably since Doupé et al.'s main concern is about the crawling phase of the scanner, whereas we focus on the attack phase, and specifically on the payloads.

5.1 Payload Extraction Results

The number of extracted payloads for all scanners is shown in Fig. 1(a).

Since the detection rate is approximately the same for all scanners (on the first version of Firing Range, before **Phase 4 (Retrofitting Negative Payloads)**), the number of distinct payloads, shown in Fig. 1(a), is interesting: the detection technique of **Scanner 3** uses far fewer payloads, while achieving the same detection of others. The comparatively larger number of payloads observed in the first 2 scanners is due to the use of unique identifiers tied to each of requests. Arguably, these identifiers are used to link a result back to the request that originated it even if server side processing had moved it around—this property is important when detecting stored XSS.



(a) Extracted payloads. (b) Generated templates.

Fig. 1. Output summary of Phase1 and Phase2, respectively.

Recommendation (Baseline Payloads). Having a good, strong source of baseline payloads (e.g., from exploit databases[3]) makes a significant difference in terms of payload-to-detection rate. Having a diverse set of distinct payloads is better (although harder) than having a small set of initial payloads used to generate many payloads by automatic mutation or fuzzing. This does not mean that scanner designers should not mutate the baseline payloads to generate new ones. Mutation and fuzzing are fundamental, but should not substitute the research effort by the vendor, who should ensure a rich supply of exploitation payloads whenever new vulnerabilities are discovered.

5.2 Payload Templating Results

The number of payloads alone, however, does not tell much about the actual quality and type of the payloads. More interesting conclusions about the fuzzying algorithm adopted by each scanner can be drawn by comparing Fig. 1(a) vs. (b). Indeed, after applying the clustering process of Phase 2 (Payload Templating), we notice immediately the limited number of templates (i.e., reflecting the attack patterns), as shown in Fig. 1(b).

The larger number of templates generated for **Scanner 2** is an index of lower efficiency of the scanner, in terms of amount of requests and time spent for detecting a given set of vulnerabilities. In fact, while detecting the very same set of vulnerabilities as the other scanners, **Scanner 2** employs 3–4 times the number of payload *templates* (i.e., distinct exploit patterns).

At this point in the analysis we could already see some scanners emerging as clearly more efficient due to the smaller number of templates they use. For example, **Scanner 2** uses more complex payload templates such as:

```
-STR-‘"--></style></script><script>alert(0x0000-STR-_NUM_)
    </script>
```

[3] http://exploit-db.com/search/?action=search&filter_description=cross&filter_platform=0&filter_type=6.

The templates for **Scanner 4** are numerous and very different from each other, due to the wide variety of generated payloads. The (low number of) templates from **Scanner 5** show that its payloads are significantly different from the rest. The templates that cover most of the payloads are:

```
-->">'>'"<obf000084v209637>
.htaccess.aspx-->">'>'"<obf000085v209637>
.htaccess.aspx-->">'>'"<obf000_NUM_v209637>
-STR--->">'>'"<obf_NUM_v209637>
```

which capture the scanner developer's particular interest in generic payloads that can highlight the incorrect escaping of a number of special characters at once. This approach is not found in any other scanner.

Scanner 6 created a large number of templates, sign of strong use of non-trivial mutations and fuzzying.

Recommendations (Mutation and Context). From our results we learn that designing good mutation mechanism is not easy to implement. Naïve approaches such as those adopted by **Scanner 1**, which only appends a progressive number to "fuzz" the payload, do not pay back in more vulnerabilities detected. Instead, it is inefficient as it exercises the application with the very same (small) number of payload templates, which are actually all duplicate (except for the progressive number). This can be partially mitigated if the scanner implements an intelligent algorithm that figures out that N "very similar payloads" (e.g., same strings, except for 1–2 characters) are continuously generating the same (negative) result. Setting a limit on this is a simple yet effective technique to keep efficiency under control. Moreover, a contextual parsing approach is recommended to select a candidate subset of potentially successful payloads, *before* attempting the injection. Although having an identifier (e.g., incremental number) for each injected payload is useful (e.g., for matching stored XSS or multiple injection points in a single page), it should not be used alone as a fuzzying mechanism.

Table 1. Summary of template evaluation.

Scanner	Mutations (M4)	Callbacks (M3)	Filter evasion (M2, M4, M5)
1	✓		✓
2	✓	✓	✓
3	✓	✓	✓
4	✓		✓
5			
6	✓		✓

5.3 Template Evaluation Results

During this phase we evaluated each of the templates on the metrics defined in Sect. 4.3. Figure 2 reports the mean of **M1 (Length)** calculated over the number of templates produced by each scanner. This is an interesting finding, which can be interpreted in two ways. On the one side, the length of the templates is in line with the minimum length of real-world payloads required to exploit XSS vulnerabilities, which is around 30 characters [4,15], which somehow justifies the choice of the payloads. On the other hand, such a long string may fail to exploit a vulnerable entry point that simply cuts the payload off, even without a proper filter. Although this can be a good way for the scanner to avoid flagging unexploitable vulnerabilities (false positives), it has been shown that multiple small payloads can be combined to generate a full attack [9]. However, the scanners that we examined miss these occurrences.

We notice that **Scanner 1** employs significantly longer payloads than **Scanner 5**. This can be explained, considering that **Scanner 5**'s M5 is zero, meaning that it uses no known filter-evasion techniques: thus, **Scanner 5** is less sophisticated than **Scanner 1**.

Using M2–M5, we derived Table 1, which gives a bird's eye view on the use of mutations, filter evasion and use of callbacks from each scanner. Regarding mutations and callbacks, we use M3 (Custom callbacks) and M4 (Multiple encodings), respectively, whereas for filter evasion, if at least one template has a non empty character set (from M2), uses multiple encodings (from M4), and adopt at least one evasion technique (from M5) we conclude that the scanner performs filter evasion.

As it can be seen, both random mutations and filter-evasion techniques are widely employed in the scanners that we tested. Nevertheless, these techniques are ineffective at triggering all the vulnerable entry points. In fact, most of them yield poor-quality payloads, as detailed in Sect. 5.4. Instead, the use of custom callbacks over parsing or standard JavaScript functions is not common among the scanners that we tested.

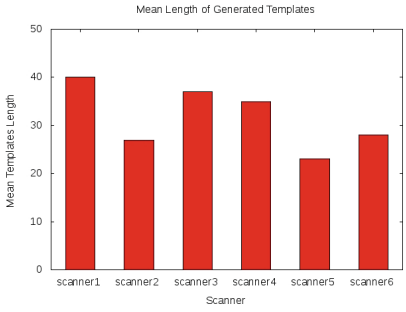


Fig. 2. Mean M1 (Length) over the templates of each scanner.

Recommendations (Payload Quality Ranking). This section answers one of the question that we stated in the introduction of this paper, which is actually one of the questions that motivated us to pursue this work: Can we rank [them] and identify a set of "better" payloads? Although the results of our experiments can be used to rank the scanners based, for example, on the quality metrics, unfortunately none of the scanners that we reviewed offer feedback to the analyst regarding the "quality" of the payloads that successfully triggered vulnerabilities. Although we can assume that it always possible for an attacker to send an arbitrarily crafted URL to the victim, certain payloads are more likely to pass undetected by network- or application-level protections (e.g., ModSecurity), which in case of legacy web applications are the only viable alternative to patching. Therefore, a successful payload with, say, a small character set, which adopts filter-evasion techniques while keeping a short length overall, should be ranked as "high risk". This additional feedback could be part of the typical vulnerability-ranking part that some vendors include in their reports.

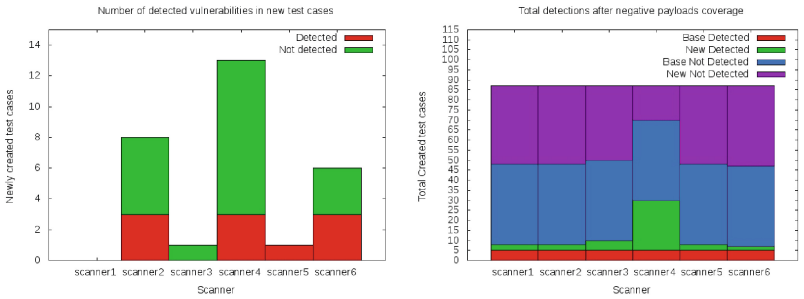
5.4 Retrofitting Negative Payloads Results

XSS Peeker's workflow iteratively adds new test cases in our testbed to account for payloads that triggered no vulnerabilities (i.e., negative payloads). This section answers one of the questions that we stated in the introduction, that is: "Is a scanner exercising all the entry points with the expected payloads?" The expected result was the reduction of the number of negative payloads to zero. So, we ran all the scanners against the new testbed and analyzed the results (note that the payloads of **Scanner 1** were all already covered by our initial test case, thus we created no additional cases).

Unfortunately, as Fig. 3(a) shows, scanners failed to detect most of the new vulnerabilities. The most surprising finding is that the very same scanners that generated a negative payload would still fail to detect and report the new test case introduced for it, even if we manually confirmed that all of the negative payloads do in fact trigger an XSS on their respective new cases.

This may indicate a bugged response-parsing routine (i.e. a functional bug in the scanner). In other cases, by manually analyzing the requests toward the new cases, we discovered that some scanners did not exercise the test case with the "right" payload: a faulty (or random) payload-selection procedure somehow failed to choose it, using it instead in test cases where it would turn out to be ineffective; a different bug class in scanners. Another interesting result is that, after the introduction of the new cases, some scanners started using payloads we had not observed before. This behavior suggests some degree of context awareness, as scanners would only generate this new set of templates after having observed these new contexts. However, even in this case we observed a staggering high rate of failures for the new corner cases we added.

These findings would have been very difficult to reveal with a static set of vulnerabilities, as opposed to the incremental approach that we adopted. Figure 3 shows the overall results produced by each scanner after including the new test cases. Although scanners did not achieve the expected results, this



(a) "New" test cases alone. (b) Overall comparison.

Fig. 3. Summary of final results after Phase 4. "New", "Base" (initial test cases), "Detected" (true positive), and "Not Detected" (false negative).

process allowed us to greatly increase our coverage of test cases for attacks supported by the analyzed scanners, and to produce a state of the art testbed for future work.

Recommendations (Continuous Testing). Although scanner vendors take testing very seriously, it is not trivial to account for the side-effect caused by adding new vulnerability cases to the testbed web applications. Adopting a retrofitting approach similar to the one that we implemented could be a first step toward finding corner cases (e.g., exercising a vulnerability with an incorrect payload) or bugs similar to the ones that we discovered.

6 Conclusions

This vertical study on XSS vulnerability scanners proposes quality metrics of 6 commercial and open-source products through passive reverse engineering of their testing phases, and manual and automated analysis of their payloads. Furthermore, we created a reusable and publicly available testbed.

By iterating on payloads that triggered no test cases, we were able to noticeably improve our test application and draw important conclusions about each scanner's inside workings. One of the key results is that, despite having some kind of awareness about context, all of the tested scanners were found wanting in terms of selecting the attack payloads and optimizing the number of requests produced. A significant number of detection failures suggest bugs and instability in the detection engines, while the high variance in types and features of the payloads we inspected makes the case for cooperation in defining common, efficient and reliable payloads and detection techniques.

References

1. Bau, J., Bursztein, E., Gupta, D., Mitchell, J.: State of the art: automated black-box web application vulnerability testing. In: IEEE SSP, pp. 332–345, May 2010. doi:10.1109/SP.2010.27
2. Doupé, A., Cova, M., Vigna, G.: Why johnny can't pentest: an analysis of black-box web vulnerability scanners. In: Kreibich, C., Jahnke, M. (eds.) DIMVA 2010. LNCS, vol. 6201, pp. 111–131. Springer, Heidelberg (2010)
3. Foundstone: Hacme Bank v2.0 (2006). http://www.foundstone.com/us/resources/proddesc/hacmebank.html
4. Gnarlysec: XSS and ultra short URLs (2010). http://gnarlysec.blogspot.ch/2010/01/xss-and-ultra-short-urls.html
5. Heiderich, M., Schwenk, J., Frosch, T., Magazinius, J., Yang, E.Z.: mXSS attacks: attacking well-secured web-applications by using innerHTML mutations. In: CCS, pp. 777–788. ACM (2013)
6. Klein, A.: DOM based cross site scripting or XSS of the third kind (2005). http://www.webappsec.org/projects/articles/071105.shtml
7. Lekies, S., Stock, B., Johns, M.: 25 million flows later: large-scale detection of dom-based XSS. In: *CCS*, pp. 1193–1204. ACM (2013)
8. Maggi, F., Frossi, A., Zanero, S., Stringhini, G., Stone-Gross, B., Kruegel, C., Vigna, G.: Two years of short urls internet measurement: Security threats and countermeasures. In: WWW, pp. 861–872 (2013)
9. Mutton, P.: XSS in confined spaces (2011). http://www.highseverity.com/2011/06/xss-in-confined-spaces.html
10. Open Web Application Security Project: Top ten (2013). https://www.owasp.org/index.php/Top_10_2013-Top_10
11. Pitsillidis, A., Levchenko, K., Kreibich, C., Kanich, C., Voelker, G.M., Paxson, V., Weaver, N., Savage, S.: Botnet judo: fighting spam with itself. In: DNSS, San Diego, California, USA. The Internet Society, March 2010
12. Samuel, M., Saxena, P., Song, D.: Context-sensitive auto-sanitization in web templating languages using type qualifiers. In: CCS, pp. 587–600. ACM (2011)
13. Dell SecureWorks: Dell SecureWorks Threat Report for 2012 (2012). http://www.secureworks.com/cyber-threat-intelligence/threats/2012-threat-reviews
14. Suto, L.: Analyzing the accuracy and time costs of web application securityscanners (2010). http://www.ntobjectives.com/files/Accuracy_and_Time_Costs_of_Web_App_Scanners.pdf
15. Toews, B.: XSS shortening cheatsheet (2012). http://labs.neohapsis.com/2012/04/19/xss-shortening-cheatsheet
16. Tudor, J.: Web application vulnerability statistics (2013). http://www.contextis.com/files/Web_Application_Vulnerability_Statistics_-_June_2013.pdf
17. Vieira, M., Antunes, N., Madeira, H.: Using web security scanners to detect vulnerabilities in web services. In: IEEE/IFIP DSN, pp. 566–571, June 2009
18. IBM X-Force: IBM X-Force 2013 Mid-Year Trend and Risk Report (2013). http://securityintelligence.com/cyber-attacks-research-reveals-top-tactics-xforce

TPM and Internet of Things

A Utility-Based Reputation Model for the Internet of Things

Benjamin Aziz[1(✉)], Paul Fremantle[1], Rui Wei[2], and Alvaro Arenas[3]

[1] School of Computing, University of Portsmouth, Portsmouth, UK
{benjamin.aziz,paul.fremantle}@port.ac.uk
[2] Department of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
12120463@bjtu.edu.cn
[3] IE Business School, IE University, Madrid, Spain
alvaro.arenas@ie.edu

Abstract. The MQTT protocol has emerged over the past decade as a key protocol for a number of low power and lightweight communication scenarios including machine-to-machine and the Internet of Things. In this paper we develop a utility-based reputation model for MQTT, where we can assign a reputation score to participants in a network based on monitoring their behaviour. We mathematically define the reputation model using utility functions on participants based on the expected and perceived behaviour of MQTT clients and servers. We define an architecture for this model, and discuss how this architecture can be implemented using existing MQTT open source tools, and we demonstrate how experimental results obtained from simulating the architecture compare with the expected outcome of the theoretical reputation model.

Keywords: Internet of things · Utility reputation · Trust management

1 Introduction

The Internet of Things (IoT) is an area where there is significant growth: both in the number of devices deployed and the scenarios in which devices are being used. One of the challenges for the Internet of Things is supporting network protocols which utilise less energy, lower bandwidth, and support smaller footprint devices. One such protocol is the MQ Telemetry Transport (MQTT) protocol [15], which was originally designed to support remote monitoring and Supervisory Control And Data Acquisition (SCADA) scenarios but has become popular for the IoT.

Another challenge with IoT networks is that small devices may not perform as well as needed due to a number of factors including: network outages or poor network performance due to the use of 2G or other low bandwidth networks, power outages for devices powered by batteries, deliberate vandalism or environmental damage for devices placed in public areas, and many other such challenges. Therefore we identified that a reputation model for devices connecting by

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 261–275, 2016.
DOI: 10.1007/978-3-319-33630-5_18

MQTT would be a useful construct to express consumers' (applications') trust in the behaviour and performance of these devices as well as measure the level of performance of the server aggregating data from such devices according to some predefined Service Level Agreement (SLA). In addition, we implemented the reputation model to demonstrate that it could be used in real MQTT networks.

Our model of reputation is based on the notion of a *utility function*, which formally expresses the consumer's level of satisfaction related to various issues of interest against which the reputation of some entity is measured. In the case of MQTT networks, one notable such issue is the Quality of Service (QoS) with regards to the delivery of messages; whether messages are delivered exactly once, more than once or at most once to their consumers. The model, inspired by previous works [6,19], is general enough to be capable of defining the reputation of client devices and servers at various levels of abstraction based on their level of performance in relation to the delivery of messages issue of interest.

The paper starts with an overview of the MQTT protocol (Sect. 2). From this, we then mathematically define the reputation model for MQTT clients and server (Sect. 3), based on their ability to keep to the requirements of the protocol. We then outline a system architecture (Sect. 4) for monitoring the MQTT protocol and thereby being able to calculate the reputation by observing the behaviour of MQTT clients and server in a real network. We show how this system was implemented and we demonstrate the results of this implementation (Sect. 5). Finally we look at related work (Sect. 6) and conclude the paper outlining areas for further research (Sect. 7).

2 MQTT Overview

MQTT [9] is described as a lightweight broker-based publish/subscribe messaging protocol that was designed to allow devices with small processing power and storage, such as those which the IoT is composed of, to communicate over low-bandwidth and unreliable networks. The publish/subscribe message pattern [10], on which MQTT is based, provides for one-to-many message distribution with three varieties of delivery semantics, based on the level of QoS expected from the protocol. In the "at most once" case, messages are delivered with the best effort of the underlying communication infrastructure, which is usually IP-based, therefore there is no guarantee that the message will arrive. This protocol is termed the QoS = 0 protocol. In the second case of "at least once" semantics, certain mechanisms are incorporated to allow for message duplication. Despite the guarantee of delivering the message, there is no guarantee that duplicates will be suppressed. This protocol is also known as the QoS = 1 protocol. Finally, for the last case of "exactly once" delivery semantics, also known as the QoS = 2 protocol, the published message is guaranteed to arrive only once at the subscribers. The protocol also defines message structures needed in communications between *clients*, i.e. end-devices responsible for generating data from their domain (the data source) and *servers*, which are the system components responsible for collating source data from clients/end-devices and distributing these data to interested subscribers. Servers are often also referred to as *brokers*, as they intermediate between the data publishers and subscribers.

3 A Reputation Model for MQTT

We show in this section how the model of reputation defined for business processes in [7,8] can be adapted, with minimum changes, to the MQTT protocol to obtain the reputation of client devices and the server.

3.1 Monitoring Events

Central to the model defined by [7,8] was the notion of an *event*, which is a signal produced by an independent *monitor system*, which is monitoring the interactions occurring between the different entities in the monitored environment, in this case the client and server entities participating in the MQTT protocol. An event is defined as follows:

$$Event : TimeStamp \times Ag \times Msg \times Id \times \mathbb{N}$$

where *TimeStamp* is the timestamp of the event generated by the monitor system issuing it, *Ag* is the identity of the agent (client device or server) to whom the event is related, *Msg* is the specific message of interest (e.g. *Publish* and *Pubrel* messages), *Id* is an identity value of the protocol instance and finally, $\mathbb{N}$ is a natural number representing the number of times the message *Msg* has been monitored, i.e. was sent.

For example, the following event, issued at monitor system's local time:

$$ev_{ex1} = (12{:}09{:}52, temp\_sensor, Publish, 1234, 2)$$

denotes that the *temp_sensor* device has been monitored, within the instance number 1234 of the protocol, to have sent twice the *Publish* message to the server responsible for collecting environment temperature data. On the other hand, the following event issued at local time 12:19:02:

$$ev_{ex2} = (12{:}19{:}02, temp\_server, Publish, 1234, 1)$$

denotes that the server responsible for the environment temperature, *temp_server*, has been monitored, within the same instance number 1234 of the protocol, to have published only once the specific message *Publish* to the subscribers of the temperature topic. In both these examples, the assumption is that the monitor system is capable of detecting that the protocol instance being monitored has terminated before it issues any events related to that instance. Although theoretically this is impossible due to the halting problem, in practical terms, the monitor system can assume the protocol to have terminated after some reasonable amount of time has elapsed since the last protocol message.

The monitor generates events in the above form, which are used by a *reputation engine* to determine the reputation values for client devices and servers in an MQTT-based environment. The reputation engine will then use a *utility function* pre-supplied to the engine by subscribers to determine the level of satisfaction of a subscriber with regards to the results reported within an event:

$$\begin{array}{|l} \mathit{utility} : \mathit{Event} \times \mathit{SLA} \rightarrow [0,1] \\ \hline \forall (t,a,m,i,n) \in \mathit{Event}, \mathit{sla} \in \mathit{SLA} \bullet \mathit{utility}((t,a,m,i,n),\mathit{sla}) = r \in \mathbb{R} \end{array}$$

This utility function will consider a SLA, defined as follows:

$$\begin{array}{|l} \mathit{SLA} : \mathit{Ag} \times \mathit{Top} \times \mathit{Iss} \rightarrow \mathbb{N}^0 \end{array}$$

Here the SLA considers an issue of interest to the subscriber, *Iss*, which will be in our case the QoS level value fixed to one of 0, 1 or 2, expected from a particular agent *Ag* in relation to a specific topic *Top*. The outcome of the utility function is a real number r representing the satisfaction level of the subscriber in terms of both the SLA and the real values reported by events.

For example, consider the following SLA instance

$$\mathit{sla} = ((\mathit{temp\_server}, \mathit{temperature}, \mathit{QoS}), 2)$$

then given the event ev_{ex2}, the utility function could return the following value:

$$\mathit{utility}(\text{t,temp\_server,Publish,1234,1,((temp\_server,}\mathit{temperature}\text{,QoS),2)}) = 1$$

This indicates that the subscriber's requirements have been fulfilled, as indicated by their SLA ($r = 1$), with the results reported by the event ev_{ex2}. On the other hand, considering the same SLA, the utility function might return:

$$\mathit{utility}(\text{t,temp\_server,Publish,1234,0,((temp\_server,}\mathit{temperature}\text{,QoS),2)}) = 0$$

to show that the subscriber has a satisfaction value of 0 since the number of times the message was delivered to the subscriber is lower (i.e. 0) than what its QoS level is defined in the SLA (i.e. 2), therefore breaching the exactly-once delivery semantics to the subscriber principle in MQTT.

Since the number of times a message is delivered will either confirm or not to the level of QoS expected by the subscriber, in all of the above cases, the score given will reflect either total satisfaction (i.e. 1) or total dissatisfaction (i.e. 0).

3.2 Reputation Models

After introducing the main notions of an event and a utility function, we can now define models of reputation for the clients (e.g. sensor devices) and the MQTT server (broker) that aggregates the messages from the clients before publishing them to the subscribers. The subscribers are assumed to be the business applications or data consumers, and we do not include them in the reputation model. The MQTT standard does not prohibit a client from acting as both a device (i.e. source of data) and a subscriber (i.e. consumer of data). However, in our case, we only measure the reputation of the "source of data" clients.

The Server Reputation Model. The first reputation model reflects the behaviour of MQTT servers. Given a set of events, *Event*, captured by the monitor system and relevant to the server for whom the reputation is being calculated, then we can define the server's reputation function computed at a particular point in time and parameterised by a specific SLA as follows:

$$[Srv, SLA, TimeStamp]$$
$$s\_rep\_sla : Srv \times SLA \times TimeStamp \rightarrow [0, 1]$$
$$\forall\, eset_s : \wp(Event) \bullet$$
$$s\_rep\_sla(s, sla, t) = \frac{\sum \varphi(t, te)\, utility(ev, sla)}{\#eset_s}$$

where $\#s$ denotes the cardinality of a set s and $\varphi(t, te)$ is a time discount function that puts more importance (emphasis) on events registered closer in time to the moment of computing the reputation. One definition of $\varphi(t, te)$ could be the time discount function defined by [13], which we redefine here as $\varphi(t, te) = e^{-\frac{t-te}{\lambda}}$, where t is the current time at which the reputation is calculated, te is the timestamp of the event being considered and λ is *recency scaling factor* used to adjust the value of the function to a scale required by the application. After this, the server reputation function, *s_rep_sla*, is defined as the weighted average of the utilities obtained from all the generated events with respect to some SLA.

The above definition aggregates the set of all relevant events, i.e. the events that first have the same server name as that appearing in the SLA and second that are on an instance of the protocol related to the topic of the SLA. The first condition is checked using the two operators *fst* and *snd*, which will return the first and second elements of a tuple, whereas the second condition is checked using the predicate $id\_top(ev, sla)$, which returns a True outcome if and only if the identity number of an instance of a protocol captured by ev corresponds to the topic value mentioned in the SLA *sla*. Considering the example events of the previous section, we would have the following calculation of $id\_top(ev, sla)$:

$id\_top(12{:}09{:}52, temp\_server, Publish, 1234, 1, ((temp\_server, temperature, QoS), 2)) = True$

The above definition calculates the sum of the time-discounted utility function values, with respect to the given SLA and the events gathered, and average these over the total number of events gathered ($\#eset_s$) in any one instance when this reputation value is calculated.

Based on the definition of *s_rep_sla*, we next aggregate the reputation of a server across every SLA that binds that server to its subscribers:

$$[Srv, TimeStamp]$$
$$s\_rep : Srv \times TimeStamp \rightarrow [0,1]$$
$$\forall\, slaset_s : \wp(SLA) \bullet s\_rep(s,t) = \frac{\sum_{sla \in slaset_s.fst(sla) = s} s\_rep\_sla(s,sla,t)}{\#slaset_s}$$

Which provides a more general indication of how well a server s behaves in relegation to a number of subscribers. This reputation is again calculated in a particular point in time, t, however it is straightforward to further generalise this reputation function over some time range, between t and t'.

The Client Device Reputation Model. After introducing the reputation model of the server, we define here the client's reputation model. Like the server, a client might also be implementing the QoS correctly, but it requires multiple reconnections, duplicate messages etc., while the server does not. For instance, if the devices are not sending PINGs or responding to them, or this is delayed, it might indicate a problem is more likely to occur in the future. Similarly, if the device needs to send multiple duplicate messages or needs to be sent duplicate messages, it also might indicate possible failure in the future. Thus, the reputation model for a client may be based on either the "Keep Alive/PING" case or the "Client's Retransmission Procedure" case. However, we start with the definition of an overall reputation model that generalises these two cases.

Given a set of events, *Event*, captured by the monitor system relevant to some client, then we define the client's reputation function computed at a particular point in time in a specific process (Keep Alive/PING procedure or retransmission procedure) and parameterised by a specific SLA as follows:

$$[Client, SLA, TimeStamp, Procedure]$$
$$c\_rep\_sla\_p : Client \times SLA \times TimeStamp \times Procedure \rightarrow [0,1]$$
$$\forall\, pset_s : \wp(Event) \bullet c\_rep\_sla\_p(c, sla, t, p) = \frac{\sum_{ev \in pset_s.snd(ev) = fst(sla) = c \,\wedge\, id\_top(ev,sla)} \varphi(t,te)utility(ev,sla)}{\#pset_s}$$

This definition gathers the set of all related events, i.e. the events that first have the same client name as that appearing in the SLA and second that are on an instance of the protocol related to the topic of the SLA. The definition is parameterised by the client, an SLA, a timestamp and the specific procedure (e.g. Keep Alive/PING or retransmission). The SLA represents what the expectation is, from the server's point of view, of the client's behaviour in the context of the specific procedure. Similar to the case of *s_rep_sla*, a utility function is applied to measure the satisfaction of the server, in a time-discounted manner, in relation to the client's behaviour and this is then averaged over the total number of events captured in a specific instance of time.

For example, consider the case of the Keep Alive/PING procedure, then *c_rep_sla_ka* is defined as the time-discounted average of the utilities obtained from all generated events with respect to the Keep Alive/PING procedure.

$$[Client, SLA, TimeStamp, KeepAlive]$$

$$c\_rep\_sla\_ka : Client \times SLA \times TimeStamp \times KeepAlive \rightarrow [0, 1]$$

$$\forall\, pset_s : \wp(Event) \bullet c\_rep\_sla\_ka(c, sla, t, ka) = \frac{\sum_{ev \in kapingset_s.snd(ev) = fst(sla) = c \,\wedge\, id\_top(ev,sla)} \varphi(t,te)utility(ev,sla)}{\#kapingset_s}$$

In this procedure, the client sends a Pingreq message within each KeepAlive time period, then the receiver answer with a Pingresp message when it receives a Pingreq message from the gateway to which it is connected. Clients should use KeepAlive timer to observe the liveliness of the gateway to check whether they are connected to broker. If a client does not receive a Pingresp from the gateway even after multiple retransmissions of the Pingresq message, it fails to connect with gateway during the Keep Alive period.

Hence, for the above example, using *id_top* to show a set of related events *ev* corresponds to the topic value mentioned in the SLA, *sla*, we would have that:

$id\_top(12:09:52, client, Pingreq, False, False, 1234, 1, ((client, temperature, QoS), 0)) = True$

The event $ev_{kaping} = (12:09:52, client, Pingreq, False, False, 1234, 1)$ generated by the monitor, could reflect a client device that has sent once the Pingreq message to connect to the gateway within the instance number 1234 of the protocol during the Keep Alive period. Then, given the SLA instance $sla = ((client, temperature, QoS), 0)$, the client should deliver this Pingreq message in relation to a specific topic (in this case *temperature*) at most once within each KeepAlive time period, but there is no guarantee the message will arrive.

From the definition of *c_rep_sla_p*, we generate a more general reputation for some client in a particular point in time t within a period, *Period*, as follows:

$$[Client, SLA, TimeStamp]$$

$$c\_rep\_sla : Client \times SLA \times TimeStamp \rightarrow [0, 1]$$

$$\forall\, periodset_s : \wp(Period) \bullet c\_rep_s la(c, sla, t) = \frac{\sum_{ev \in periodset_s.snd(ev) = fst(sla) = c \,\wedge\, id\_top(ev,sla)} c\_rep\_sla\_p(c,sla,t,p)}{\#periodset_s}$$

Giving an example based on the Keep Alive/PING procedure, assume the KeepAliveTimer is set to 60, then calculating $c\_rep\_sla(c, sla, t)$ will give us

the reputation of the client device during the whole Keep Alive period of 60 seconds. In another example, based on the retransmission procedure, we assume that Nretry is set to 10. Aggregating over the $c\_rep\_sla(c, sla, t)$ values yields reputation in relation to the client's retransmissions within a 10 time-unit limit.

Finally, based on the definition of *c_rep_sla*, we can further generalise the reputation value over all relevant SLAs for a specific client, c, and in a particular point in time, t, as follows:

$$[Client, SLA]$$
$$c\_rep : Client \times SLA \rightarrow [0, 1]$$
$$\forall\, slaset_s : \wp(SLA) \bullet c\_rep(c, t) = \frac{\sum_{c_{rep}(c,t) = sla \in slaset_s . fst(sla) = c} c\_rep\_sla(s, sla, t)}{\#slaset_s}$$

This definition gives a more general indication of how well the client device generally behaves in relation to the SLAs it holds with the server (possibly on behalf of the subscribers dealing with the server). These could include scenarios where the clients might use the Keep Alive/PING procedure to observe the liveliness of the gateway to check whether they are connected to a broker. Moreover, in the case of messages that expect a response, if the reply is not received within a certain time period, the client will be expected to retransmit this message.

The reputation model of a client in different procedures might cause different failures. Thus, as we demonstrated above the first reputation model, *c_rep_sla_p*, will lead to new models with slight variations capturing this variety of failures. For example, for the case of a client's retransmission procedure, all messages that are "unicast" to the gateway and for which a gateway's response is expected are supervised by a retry timer *Tretry* and a retry counter *Nretry*. The retry timer *Tretry* is started by the client when the message is sent and stopped when the expected gateway's reply is received. If the client does not receive the expected gateway's reply during the *Tretry* period, it will retransmit the message. In addition, the client should terminate this procedure after *Nretry* number of retransmissions and should assume that it is disconnected from the gateway. The client should then try to connect to another gateway only if it fails to re-connect again to the previous gateway.

One such client reputation, is defined based on a specific *TRetry* timer:

$$[Client, SLA, TimeStamp, TRetry]$$
$$c\_reptr\_sla\_tr : Client \times SLA \times TimeStamp \times TRetry \rightarrow [0, 1]$$
$$\forall\, tretryset_s : \wp(Event) \bullet c\_reptr\_sla\_tr(c, sla, t, tr) = \frac{\sum_{ev \in kapingset_s . snd(ev) = fst(sla) = c \,\wedge\, id\_top(ev, sla)} \varphi(t, te) utility(ev, sla)}{\#tretryset_s}$$

For example, consider the following $sla = ((client, temperature, QoS), 1)$, then given the event $ev_{tretry} = (12 : 09 : 52, client, Publish, False, True, 1234, 2)$, it could reflect an event in the retransmission procedure. If the client does not receive a Puback message with QoS level 1 within a time period defined by the *TRetry* value, the client may resend the Publish message with the DUP flag set. When the server receives a duplicate message from the client, it re-publishes the message to the subscribers, and sends another Puback message.

Similarly, another variation of the client's reputation function may be based on the NRetry counter instead:

$$
\begin{array}{|l}
\hline
[Client, SLA, TimeStamp, NRetry] \\
c\_repnr\_sla\_nr : Client \times SLA \times TimeStamp \times NRetry \rightarrow [0,1] \\
\hline
\forall\, nretryset_s : \wp(Event) \bullet c\_reptr\_sla\_tr(c, sla, t, nr) = \\
\dfrac{\sum\limits_{ev \in kapingset_s.snd(ev) = fst(sla) = c \;\wedge\; id\_top(ev, sla)}^{\varphi(t,te)utility(ev,sla)}}{\#nretryset_s} \\
\hline
\end{array}
$$

Again, for the above definition, for $sla = ((client, temperature, QoS), 2)$, and given the event $ev_{nretry} = (12 : 09 : 52, client, Publish, False, False, 1234, 1)$, it could indicate that the client should not retransmit again in the retransmission period due to the fact that QoS is set to 2 (meaning the message is guaranteed to be delivered exactly-once to the subscribers). In this case, the DUP flag must be set to False, in order to prevent a retransmission.

4 A Reputation System Architecture for MQTT

Our architecture for a reputation system for an MQTT network is composed of a *reputation monitor* and a *reputation engine*, as shown in Fig. 1.

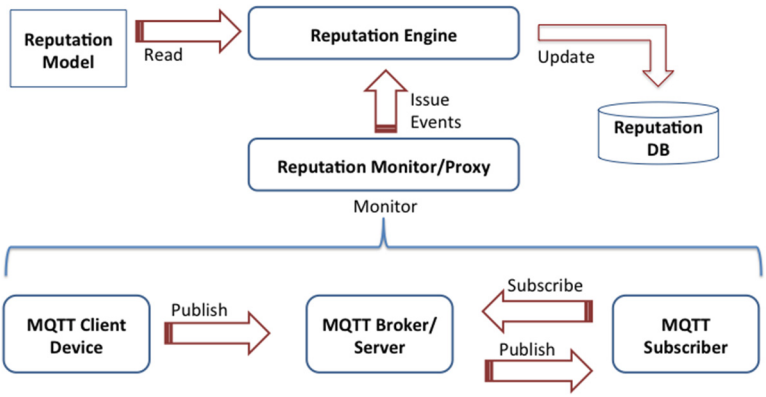


Fig. 1. The reputation system architecture.

The architecture defines the capabilities of the various components in an MQTT network. The reputation monitor (also sometimes referred to as the proxy) will *monitor* the MQTT interactions that take place among the MQTT network components, namely the client devices, server and subscribers. Monitoring implies that the reputation monitor will issue events to the reputation engine whenever these are required after each time it has captured an MQTT communication relevant to the utility functions predefined by the consumers (possibly the subscribers). These events could represent aggregations/abstractions of data collected from such communications, in order to minimise the additional network traffic created by this process.

Once an event has arrived at the reputation engine, it is either stored for applying further aggregations/abstractions or it is used immediately to compute new updates for the various reputation values for the clients and the server. The calculations are based on the reputation models defined in the previous sections, and the updates to these reputation values are then stored in a local reputation database. In our architecture, we only consider the monitoring problem, however, it is easy to extend this architecture in the future to include a control step, where the reputation values for different participants are then used to impact/feed back into the MQTT network communications.

5 Simulation of the Model

We implemented the architecture, described in the previous section, by running a number of off-the-shelf open source tools. First, we used the Mosquitto tool [1] as the broker (server). Mosquitto is an open source MQTT broker written in the C language that implements the MQTT protocol version 3.1. The Mosquitto project is highly portable and runs on a number of systems, which made it an effective choice for our experiments.

In order to simulate the client, we used the source code for the Eclipse Paho MQTT Python client library [2], which comes with Java MQTT client API and implements versions 3.1 and 3.1.1 of the MQTT protocol. This code provides a client class, which enables applications to connect to an MQTT broker to publish messages, receive published messages and to subscribe to topics. It also provides some helper functions to make publishing one off messages to an MQTT server very straightforward. Using this library we created a set of programs that would publish and subscribe to the Mosquitto broker. Finally, to implement the monitoring function we needed to capture all the traffic between the client and the server. For this we extended the Paho MQTT *test proxy* [2], which acts as a "reverse proxy", impersonating an MQTT server and sending all the traffic on to a real broker after capturing the messages. The proxy represents a mediator between clients and the broker. By extending this proxy we were able to trace all the packets being sent and received and send monitoring information to our reputation engine in order to calculate the reputation of the client and broker.

5.1 Results

To begin with, we assume that the network will misbehave with regards to the messages that are exchanged among the various entities in the system. This misbehaviour is modelled as the network dropping some messages according to a predefined rate (e.g. 0–100 %). There could be other sources of network misbehaviour, such as the insertion of new messages and the repetition or modification of transmitted messages, however, for simplicity, we consider only the suppression of messages as our example of how the network could misbehave and how such misbehaviour would affect the reputation of MQTT clients and servers.

In our case, we chose the rate of successful message delivery to be in the range of 50 % to 100 %, where 50 % means that one message in every two is dropped by the network, and 100 % means that every message is delivered successfully to its destination. This latter case is equivalent to the normal behaviour discussed above. There are a number of tools that can drop network packets selectively. However, we created a new tool based on the above-mentioned proxy that specifically targets disrupting MQTT flows by dropping MQTT packets. The tool allowed us to target a percentage of dropped packets and therefore calculate the reputation under a given percentage of packet loss.

Since our aim is to demonstrate, in general terms, how reputation-based trust can be obtained in an IoT system such as an MQTT network, and for simplicity, we opted to consider only one source of misbehaviour, namely message suppression, without considering the other sources. Despite the fact that such sources are also interesting, they do not affect the generality of our approach.

5.2 Reputation Results

To compute the reputation value of clients and servers, we collected events related to the QoS level agreed between the client and the server throughout the 5-minute measurement window, and used the server and client reputation model proposed in Sect. 3.2 to calculate their reputation values. The QoS level monitoring is important as it is directly related to the issue of message suppression when messages are communicated over the unreliable network. In the presence of such abnormal behaviour, the reputation values of the clients and the server are shown in Fig. 2 versus the rate of successful message delivery (0.5 to 1).

From this figure, we note that despite starting at low reputation levels in line with the low delivery rate of messages, these reputation values will increase reaching the optimal value of 1 when the rate of delivery of messages is 1. This optimal case represents the case of normal behaviour when every message is delivered successfully to its destination.

6 Related Work

Reputation is a general concept widely used in all aspects of knowledge ranging from humanities, arts and social sciences to digital sciences. In computing

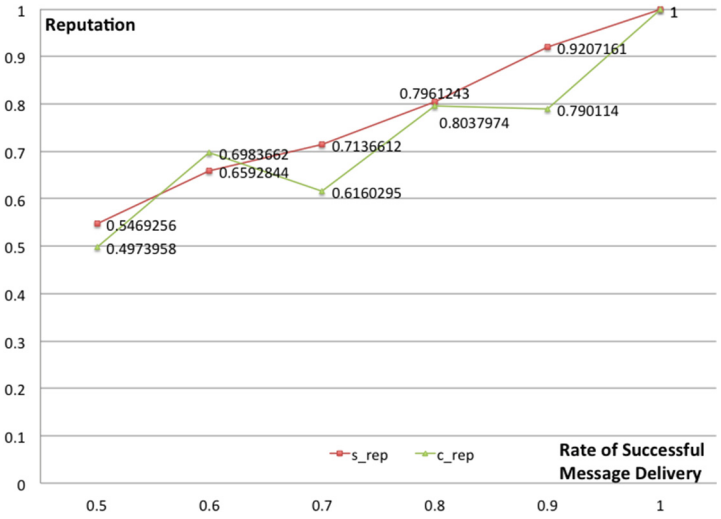


Fig. 2. Reputation values for the clients (c_rep) and servers (s_rep) vs. the rate of successful message delivery.

systems, reputation is considered as a *measure* of how trustworthy a system is. There are two approaches to trust in computer networks: the first involves a "black and white" approach based on security certificates, policies, etc. For example, SPINS [17], develops a trusted network. The second approach is probabilistic in nature, where trust is based on reputation, which is defined as a probability that an agent is trustworthy. In fact, reputation is often seen as one measure by which trust or distrust can be built based on good or bad past experiences and observations (direct trust) [14] or based on collected referral information (indirect trust) [5].

In recent years, the concept of reputation has shown itself to be useful in many areas of research in computer science, particularly in the context of distributed and collaborative systems, where interesting issues of trust and security manifest themselves. Therefore, one encounters several definitions, models and systems of reputation in distributed computing research (e.g. [12,14,20]).

There is considerable work into reputation and trust for wireless sensor networks, much of which is directly relevant to IoT trust and reputation. The Hermes [22] and E-Hermes [23] systems utilise Bayesian statistical methods to calculate reputation based on how effectively nodes in a mesh network propagate messages including the reputation messages. Similarly TRM-IoT [11] evaluates reputation based on the packet-forwarding trustworthiness of nodes, in this case using fuzzy logic to provide the evaluation framework. Another similar work is CORE [16] which again looks at the packet forwarding reputation of nodes.

Our approach differs from the existing research in two regards: firstly, the existing reputation models for IoT utilise the ability of nodes to operate in consort as the basis of reputation. While this is important in wireless sensor

networks, there are many IoT applications that do not utilise mesh network topologies and therefore there is a need for a reputation model that supports client-server IoT protocols such as MQTT. Secondly, the work we have done evaluates the reputation of a reliable messaging system based on the number of retries needed to successfully transmit a message. Although many reputation models have been based on rates of packet forwarding, the analysis of a reliable messaging system (like MQTT with QoS >1) is different as messages are always delivered except in catastrophic circumstances. Therefore we looked at the effort and retries required to ensure reliable delivery instead. We have not seen any similar approach to this and consider this the major contribution of the paper.

7 Conclusion

To conclude, we defined in this paper a model of reputation for IoT systems, in particular, for MQTT networks, which is based on the notion of utility functions. The model can express the reputation of client and server entities in an MQTT system at various levels, and in relation to a specific issue of interest, in our case the QoS level of the delivery of messages in the presence of a lossy network. We demonstrated that it is possible, using off-the-shelf open source MQTT tools, to implement an architecture of the reputation system that monitors the MQTT components, and we showed that the experimental results obtained from running such a system validate the theoretical model.

Future work will focus on adapting the reputation model and its architecture and implementation to other IoT standards, e.g. the Advanced Message Queuing Protocol (AMQP) [21], the Extensible Messaging and Presence Protocol (XMPP) [4], the Constrained Application Protocol (CoAP) [18] and the Simple/Streaming Text Oriented Messaging Protocol (STOMP) [3]. We also plan to consider other issues of interest when calculating reputation where satisfaction is not necessarily a binary decision, for example, the quality of data generated by client devices and the quality of any filtering, aggregation or analysis functions the server may apply to such data in order to generate new information to be delivered to the consumers. Further, we intend to apply Bayesian statistics to the results to improve the probabilistic calculation of the reputation values.

Some other interesting, though more advanced areas of research, include the strengthening of the model to be able to cope with malicious forms of client and server behaviour, for example, collusion across such entities in order to produce fake reputation values for a targeted victim, and a study on the welfare of IoT ecosystems based on the different rates of the presence of ill-behaved and well-behaved entities in the ecosystem, and how variations in the presence ratio of such entities would lead to a notion of reputation reflecting the wider ecosystem.

References

1. Mosquitto: An open source mqtt v3.1/v3.1.1 broker. http://mosquitto.org/. Accessed 11 Mar 2016
2. Paho. http://www.eclipse.org/paho/. Accessed 11 Mar 2016
3. STOMP: The Simple Text Oriented Messaging Protocol. https://stomp.github.io. Accessed 11 Mar 2016
4. XMPP Standards Foundation. http://xmpp.org. Accessed 11 Mar 2016
5. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: HICSS 2000: Proceedings of the 33rd Hawaii International Conference on System Sciences, vol. 6. IEEE Computer Society, Washington, DC (2000)
6. Arenas, A.E., Aziz, B., Silaghi, G.C.: Reputation management in collaborative computing systems. Secur. Commun. Netw. **3**(6), 546–564 (2010)
7. Aziz, B., Hamilton, G.: Reputation-controlled business process workflows. In: Proceedings of the 8th International Conference on Availability, Reliability and Security, pp. 42–51. IEEE CPS (2013)
8. Aziz, B., Hamilton, G.: Enforcing reputation constraints on business process workflows. J. Wirel. Mob. Netw. Ubiquit. Comput. Dependable Appl. (JoWUA) **5**(1), 101–121 (2014)
9. Banks, A., Gupta, R.: MQTT Version 3.1.1 (2015)
10. Birman, K., Joseph, T.: Exploiting virtual synchrony in distributed systems. SIGOPS Oper. Syst. Rev. **21**(5), 123–138 (1987)
11. Chen, D., Chang, G., Sun, D., Li, J., Jia, J., Wang, X.: Trm-iot: A trust management model based on fuzzy reputation for internet of things. Comput. Sci. Inf. Syst. **8**(4), 1207–1228 (2011)
12. Fullam, K., Barber, K.: Learning trust strategies in reputation exchange networks. In: AAMAS 2006: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1241–1248. ACM Press (2006)
13. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. Auton. Agent. Multi-Agent Syst. **13**(2), 119–154 (2006)
14. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. **43**(2), 618–644 (2007)
15. Locke, D.: MQ Telemetry Transport (MQTT) V3.1 Protocol Specification (2010)
16. Michiardi, P., Molva, R.: Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In: Jerman-Blažič, B., Klobučar, T. (eds.) Advanced Communications and Multimedia Security. IFIP, vol. 100, pp. 107–121. Springer, Heidelberg (2002)
17. Perrig, A., Szewczyk, R., Tygar, J., Wen, V., Culler, D.E.: Spins: Security protocols for sensor networks. Wirel. Netw. **8**(5), 521–534 (2002)
18. Shelby, Z., Hartke, K., Bormann, C.: Constrained Application Protocol (CoAP) draft-ietf-core-coap-18. https://tools.ietf.org/html/draft-ietf-core-coap-18. Accessed 11 Mar 2016
19. Silaghi, G.C., Arenas, A., Silva, L.: Reputation-based trust management systems and their applicability to grids. Technical report. TR-0064, Institutes on Knowledge and Data Management & System Architecture, CoreGRID - Network of Excellence, February 2007. http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0064.pdf

20. Silaghi, G.C., Arenas, A., Silva, L.M.: Reputation-based trust management systems and their applicability to grids. Technical report TR-0064, Institutes on Knowledge and Data Management and System Architecture, CoreGRID - Network of Excellence, February 2007
21. Vinoski, S.: Advanced message queuing protocol. IEEE Internet Comput. **10**(6), 87–89 (2006)
22. Zouridaki, C., Mark, B.L., Hejmo, M., Thomas, R.K.: Hermes: A quantitative trust establishment framework for reliable data packet delivery in manets. J. Comput. Secur. **15**(1), 3–38 (2007)
23. Zouridaki, C., Mark, B.L., Hejmo, M., Thomas, R.K.: E-hermes: A robust cooperative trust establishment scheme for mobile ad hoc networks. Ad Hoc Netw. **7**(6), 1156–1168 (2009)

Advanced Remote Firmware Upgrades Using TPM 2.0

Andreas Fuchs(✉), Christoph Krauß, and Jürgen Repp

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
{andreas.fuchs,christoph.krauss,juergen.repp}@sit.fraunhofer.de

Abstract. A central aspect for securing connected embedded systems are remote firmware upgrades to deal with vulnerabilities discovered after deployment. In many scenarios, Hardware Security Modules such as the Trusted Computing Group's Trusted Platform Module (TPM) 2.0 are used as a security-anchor in embedded systems. In this paper, we discuss the benefits of TPM 2.0 for securing embedded systems and present a concept for advanced remote firmware upgrade of an embedded system with enforcement of Intellectual Property Rights and Privacy protection of device-resident data (i.e., data that remains on the device during the flashing process). This concept utilizes unique features of TPM 2.0. Furthermore, a prototypical implementation using a hardware TPM 2.0 and the TPM Software Stack 2.0 low-level System API is presented as a proof-of-concept.

1 Introduction

Information Technology (IT) is one of the main drivers for innovations in our everyday private and work life. Especially, the use of highly connected embedded systems is increasingly growing, e.g., to enable new (safety-critical) applications in areas such as automotive, aerospace, energy, healthcare, manufacturing, or entertainment. An example is a connected car which communicates with other cars or the infrastructure to increase traffic safety and efficiency.

However, the increased connectivity (even to the Internet) and physical accessibility of the embedded systems also introduces new threats with regard to IT security and privacy. Successful attacks may have serious consequences - even to life and limb of people.

One approach for securing embedded systems is the use of Hardware Security Modules (HSMs) as a security-anchor. An HSM provides secure storage for cryptographic keys, a secure execution environment for (cryptographic) operations, and additional mechanisms such as secure boot to ensure the integrity of the platform. One promising approach is the use of the latest version of the Trusted Platform Module (TPM) 2.0, an international open standard developed by the TCG. TPM 2.0 and the corresponding TPM Software Stack (TSS) provide a high flexibility which is of great value to cost-efficiently secure different types of embedded systems on the basis of one common standard. TPM and TSS 2.0

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 276–289, 2016.
DOI: 10.1007/978-3-319-33630-5_19

can be realized in different specialized profiles. For example, the TCG already specified a TPM 2.0 Automotive Thin Profile [22] which is suitable for small resource-constrained electronic control units (ECUs) in a car such as an engine control unit. For more powerful embedded systems, other profiles such as PC Client might be appropriate. Also the TPM 2.0 itself provides a high flexibility. In contrast to TPM 1.2, a TPM 2.0 can be realized not only as an additional hardware chip (which provides high security but also introduces high costs) but also as a System on Chip (SoC) or Firmware TPM. Thus, one general approach can be applied to different types of embedded systems.

A central aspect to ensure security is the support for remote firmware upgrades to deal with vulnerabilities discovered after deployment. A great challenge for remote upgrades is the protection of device-resident data, i.e., data that remains on the device during the flashing process. Device-resident data are usually large data sets which are not changed for the upgrade and contain intellectual property of the manufacturer or data created on the device during operation containing person-related data. It must be ensured that only new upgrades by the original firmware manufacturer can be installed and downgrade attacks or attempts to install malicious firmware upgrades are prevented.

In this paper, we first discuss the benefits of integrating TPM / TSS 2.0 in an embedded system. Next, we present a concept for advanced remote firmware upgrade of such an embedded system using the TPM 2.0 as a trust anchor for realizing an enhancement to the classical secure boot. Our concept additionally includes the enforcement of intellectual property rights (IPR) and privacy protection for device-resident data. As proof-of-concept, we implemented our concept for an automotive head unit using a hardware TPM 2.0 and the TPM Software Stack 2.0 low-level System API.

This paper is organized as follows. In Sect. 2, we present background on TPM / TSS 2.0 and related work. Our concept and prototypical implementation is presented in Sect. 3. Finally, Sect. 4 concludes the paper.

2 Background and Related Work

In the last years, the security of embedded devices has been intensively investigated. The attacks on firmware upgrades go back to hacks against consumer electronics like the Playstation Portable [1] in the mid 2000 s all the way to most recent attacks against safety critical systems such as automotive ECUs [8,16].

In order to prevent completely arbitrary firmware to run, different vendor solutions and standards have been implemented. On x86 systems UEFI introduced Secure Boot in 2013 [26]. Gaming consoles include solutions of signed software since Playstation 2 and XBox 360 [27]. IPhones and Android phones included Secure Booting features from the beginning [2,3]. Automotive ECUs provide Secure Boot via SHE and EVITA [9,10,25]. All these basic Secure Boot schemes, however, target primarily the integrity of the base firmware. The security of device-resident data during such firmware upgrade in terms of confidentiality as well as integrity is not targeted by any of these solutions; especially not the binding of these against trustworthy and fresh basic firmware images.

In order to provide the security for device-resident data in our proposed scheme, the functionalities of the Trusted Platform Module (TPM) 2.0 are used. Alternative solutions such as Intel TXT, TrustZone, Security Fuse Processor, or Secure Elements [4,6,11], can usually not directly provide these feature without further extensions. Intel SGX and some GlobalPlatform Trusted Execution Environment [7,12] provide similar capabilities, but only for applications running within the shielded environment, not the Rich Operating System. The means for retaining the security of the resident data during upgrade needs to be manually implemented within each application. Alternative HSMs such as TPM 1.2, SHE or Evita [10,17,25] cannot provide the specific feature that are required within this proposed concept. Specifically, Enhanced Authorization of TPM 2.0 is required in order to provide the security guarantees for this paper.

2.1 Background on TPM 2.0 and TSS 2.0

The second iteration of the Trusted Platform Module (TPM), namely the TPM 2.0 Library Specification [18] has been released by the Trusted Computing Group (TCG) in October 2014. It provides a catalog of functionalities that can be used to build TPMs for different platforms. Since then, a so-called TPM profile has been released for PC Clients [21] in January 2015, but also a TPM profile for Automotive Thin [22] in March 2015. In June 2015, the TPM 2.0 was also approved by ISO as successor to TPM 1.2 in ISO/IEC 11889:2015 [23].

Accompanying the TPM specifications, the TCG has engaged with the specification of a TPM Software Stack (TSS) 2.0 for this new generation of TPMs. It consists of multiple Application Programming Interfaces (APIs) for different application scenarios. A so-called Feature Level API has been published for review [19] in November 2014 and is currently still under development. It targets high-level application in the Desktop and Server domain. For lower-level applications, such as embedded, the so-called System API specification [24] was released as a final version in January 2015 in conjunction with the TPM Command Transmission Interface (TCTI) for IPC module abstraction. The System API and TCTI were designed such that embedded applications could leverage on TPM 2.0 functionalities whilst minimizing requirements against the platforms. As such, they can for example be used in heap-less scenarios where only stack-based memory allocation are possible.

2.2 Difference of TPM 2.0 to TPM 1.2

Compared with TPM 1.2, TPM 2.0 is a complete rewrite. Many of the new features were not compatible with the data type and function layout of TPM 1.2. The main purpose of TPM 2.0 remained to provide Roots of Trust for Storage and Reporting. In combination with a Root of Trust for Measurement, this allows the provisioning of reliable identities, securely stored secrets, and reporting and enforcement of software integrity. The new features of TPM 2.0 in comparison to TPM 1.2 include:

- Cryptographic Agility: Whilst TPM 1.2 was restricted to RSA $\leq$ 2048 bit and SHA1, TPM 2.0 merely provides placeholders for cryptographic algorithms that can be filled up with from the TCG's Algorithm Registry [20]. The PC-Client TPM Profile for example requires RSA, ECC, SHA1, SHA256 ...
- Symmetric Algorithms: TPM 1.2 was bound to use RSA for all encryptions. This was time consuming and inefficient. TPM 2.0 also supports symmetric cryptography such as AES and HMACs.
- Enhanced Authorization: TPM 1.2 was very limited on authorization mechanisms. In general, it used SHA1 hashes of passwords. In addition, some functions further had the possibility to include bindings to a single set of PCR values (such as key usage and sealing). With TPM 2.0 the concept of Enhanced Authorization was included, which allows the forming of arbitrary policy statements based on a set of policy commands. This provides a high flexibility requiring multiple factors to be fulfilled but also to allow different paths for policy fulfillment.
- Non-Volatile Memory: With TPM 2.0 the capabilities of the TPM's integrated non-volatile memory (NV-RAM) were enhanced. They can now be used as counters, bitmaps, and even extended PCRs.
- Many more enhancement were made. A lot of those are outline in [5].

2.3 Platform Configuration Registers

One of the core functionality of TPMs is the capability to store the status of the software/firmware that was started on a device. In order to do so, each software component – starting from the pre-BIOS as Root of Trust for Measurement – calculates a hash of the next component in the boot-chain and sends this hash to the TPM. These hashes are called configuration measurements and stored within the TPM's Platform Configuration Registers (PCRs). For space efficiency, those measurements are not stored in plain, but as a hash-chain, such that a PCR's value represents a hash-chain of the measurements of the boot-chain.

The TPM's PCRs can be used to report or protect the device's integrity. The most well-known case is the concept of Remote Attestation, where a signature over a selection of PCRs is transferred to a remote entity. This entity can then determine the current configuration of a device and assess its trustworthiness. Another use case is local attestation, where certain actions of the TPM – such as data sealing (cf. Sect. 2.4) – can be restricted to specific device configuration states via PolicyPCR (cf. Sect. 2.6).

2.4 Data Sealing

The Trusted Platform Module (TPM) 2.0 has the capability for sealing data similar to the TPM 1.2. The purpose of sealing is to encrypt data with the TPM and to ensure that only this TPM can unseal the data again. In order to unseal the data, an authentication secret can be provided or a policy session that follows the scheme for Enhanced Authorization (cf. Sect. 2.6) can be used.

2.5 NV-RAM Counters

A TPM comes with an internal non-volatile memory. This memory can be used to make keys of the TPM persistent but can also be allocated by applications. The set of TPM2_NV_* commands provides ways to allocate memory in so called NV-Indices and perform read and write operations.

With TPM 2.0 additional classes of NV-Indices were introduced: Extendable NV-Indices and NV-Counters. The latter can be used to define strongly monotonic 64 bit counters that may only be incremented but not directly written, reset, or decremented. In order to prevent attackers from merely freeing and reallocating an NV-counter with a smaller value, any NV-counter is initialized to the highest value that has ever been present in any NV-counter of the TPM. This way, decrementation of NV-counter is always prevented. One of the main purposes of NV-counters is the use in Enhanced Authorization via TPM2_PolicyNV against a maximal value.

2.6 Enhanced Authorization

With TPM 2.0 a new concept for authorizing functions of objects was introduced under the name Enhanced Authorization. Any object that requires authorization can either be authorized using a secret value assigned during creation (similar to TPM 1.2) or using a policy following this scheme.

Enhanced Authorization consists of a set of policy elements that are each represented via a TPM command. Currently, eighteen different policy elements exist that can be concatenated to achieve a logical *and* in arbitrary order and unlimited number. Two of these policy elements – PolicyOr and Policy Authorize – act as logical *or*. Due to implementation requirements, policy statement are, however, neither commutative nor distributive. Once defined they need to be used in the exact same order. In this paper, we use the following notation:

$$Policy_{abc} := PolicyX_1() \wedge PolicyX_2() \wedge \ldots PolicyX_n()$$

where $Policy_{abc}$ is the "name" for this policy, such that it can be referred to from other places and $PolicyX_i()$ describes the n concatenated TPM2 Policy commands that are required to fulfill this policy.

Out of the eighteen currently defined Policy commands of the TPM 2.0 library specification, the following four policies are utilized in our concept described in Sect. 3. They are now explained in more detail.

TPM2_PolicyOr. The PolicyOr element allows the definition of logical *or* of up to eight policy branches. Each of these policy branches itself can be an arbitrary combination of policy elements of unlimited length. In order to satisfy a PolicyOr, one of the branches needs to be satisfied before calling the PolicyOr command. To add further branches, multiple PolicyOr elements can be combined. In this paper, we represent a PolicyOr as

$$Policy_{abc} := PolicyOr([Branch_1] \vee [Branch_2] \vee \ldots)$$

TPM2_Policy Authorize. PolicyAuthorize allows the activation of policies *after* the definition of an object. In order to achieve this, a public key K^{pub} is registered with a policy. This policy element then acts as a placeholder for any other policy branch that is signed with the corresponding private key K^{priv}. In order to satisfy such a policy, a branch needs to be satisfied first, and then PolicyAuthorize is called with the cryptographic signature that authorizes this branch as allowed. If such a signature for the currently satisfied branch can be provided, then the PolicyAuthorize element is also fulfilled. Logically, a PolicyAuthorize can be viewed as an *or* for all branches that were signed with the corresponding private key.

$$Policy_{abc} := PolicyAuthorize(K^{pub}_{abc})$$
$$Authorization_1 := Sig(K^{priv}_{abc}, [Branch_1])$$
$$Authorization_2 := Sig(K^{priv}_{abc}, [Branch_2])$$

After signatures have been provided for certain branches, we say:

$$Policy_{abc} := PolicyAuthorize(K^{pub}_{abc}, [Branch_1] \vee [Branch_2])$$

TPM2_PolicyPCR. The PolicyPCR element allows functions on an object inside the TPM to be restricted to a specific combination of PCR values and thereby software configurations. A similar but less flexible capability also existed with TPM 1.2 with the keyUsageInfo for keys and digestAtRelease for sealed data. With TPM 2.0 this capability can now be applied to any authorization policy. It can further be targeted at certain operations only. In combination with PolicyOr and PolicyAuthorize it is possible to also authorize several software versions.

$$Policy_{abc} := PolicyPCR(SoftwareVersion)$$

TPM2_PolicyNV. The PolicyNV element provides the possibility to include NV-indices into the evaluation of a policy. This can be used to switch between different modes of operation, by selectively enabling and disabling certain branches of a PolicyOr and PolicyAuthorize. Furthermore, it can be used to invalidate branches of a PolicyOr and PolicyAuthorize when combined with an NV-counter, as this paper presents.

$$Policy_{abc} := PolicyNV(NVindex, operation, value)$$

An example could be the comparison of the NV index NV_{abc} against a maximum allowed value of 20. If and only if the number stored within this NV_{abc} is smaller or equal to 20 can this policy element be fulfilled. Otherwise, it would fail.

$$Policy_{abc} := PolicyNV(NV_{abc}, \leq, 20)$$

3 Remote Firmware Upgrades Retaining IPR and Privacy

A Trusted Platform Module 2.0 can be used for a multitude of application cases. One of those cases is the realization of secure remote firmware upgrades with the protection of IPR-related data material and privacy sensitive information stored in device-resident data. The remainder of this paper will focus on this use case.

3.1 Scenario

Firmware upgrades are a necessity of all software based systems to fix bugs and vulnerabilities. During such a software upgrade, not all data stored on a device shall be replaced by the incoming firmware upgrade. The two categories of data not included in a software upgrade are large unchanged data sets and data created on the device during operation.

The requirements that come along with this is that consecutive software versions need to have access to this data, whilst it needs to be stored inaccessible to malicious firmware images. Also this data must become inaccessible to old versions of the firmware after an upgrade in order to prevent so called "downgrade attacks". In a *downgrade attack*, an attacker installs an outdated version of the firmware that has known vulnerabilities in order to exploit them. Such attacks have for example been used for breaking the first generation of PlayStation Portable. The classic realizations of sealing, as employed by TPM 1.2 [17], however, is not applicable in these scenarios, because they do not allow the "updating" of referenced PCR values from the sealed blobs.

Classically, this would have to be done by allowing the updater to "reseal" the data for the state after upgrading, which requires the upgrade module to be privileged. Furthermore, it was impossible to disallow usage of the old seal for accessing the data. With the framework for "Enhanced Authorization" (EA) in TPM 2.0, it is possible to achieve this use case respecting the circumstances and requirements outlined above. In the following, the set of requirements is listed, the concept described, and a prototypical implementation outlined.

3.2 Requirements

The requirements for securing large data sets and personal information during a firmware upgrade can be summarized as follows:

1. Provide confidentiality of IPR and user data.
2. Only allow original manufacturer firmware to read/write data.
3. Prevent access by old firmware after an upgrade to new firmware.
4. Do not require "privileged mode" such as updaters to unseal the data.

3.3 Concept

Our concept for providing these requirements can be divided into three phases: Provisioning, Firmware Upgrade, and Firmware Usage.

Provisioning. After provisioning of the device, the very first thing should be the definition of an NV index that represents the device's currently required minimal firmware version. This has to be done first, in order to ensure that the NV index is initialized with a value of 0 (read Zero), cf. Sect. 2.5 for details. This counter is initialized as a single 64 bit unsigned integer value and then incremented once in order to be readable:

$$NV_{Version} := TPM2\_NV\_DefineSpace(counter)$$
$$TPM2\_NV\_Increment(NV_{Version})$$

The IPR and user data is stored in an encrypted container or partition on the devices flash that lays outside of the actual firmware binaries. The key to this encrypted container is then sealed with the TPM to the following policy:

$$Policy_{Seal} := TPM2\_PolicyAuthorize(K^{pub}_{Manu})$$

This policy allow the manufacturer as the owner of K^{priv}_{Manu} to issue new policies in the future for accessing the stored data. For the initial firmware of version $v1$, the manufacturer will provide the following policy:

$$Sig\big(K^{priv}_{Manu}, \big[TPM2\_PolicyPCR(Firmware_{v1}) \wedge TPM2\_PolicyNV(NV_{Version} \leq v1)\big]\big)$$

The PCRs that represent the integer state of $Firmware_{v1}$ may be used until the nv index $NV_{Version}$ that represents the currently required minimal version exceeds the value of $v1$.

$$Policy_{Seal} := TPM2\_PolicyAuthorize\big(K^{pub}_{Manu}, \big[TPM2\_PolicyPCR(Firmware_{v1}) \wedge TPM2\_PolicyNV(NV_{Version} \leq v1)\big]\big)$$

Note that the signed policy cannot be part of those components of the firmware that is measured into the representing PCRs. The reason is that this would lead to a cyclic relation that cannot be fulfilled.

Firmware Upgrade. Whenever a firmware upgrade is issued by the manufacturer, it will be accompanied by a newly signed policy, that (similar to the original policy) grants access to the encrypted container based on the PCR representation of that firmware. This access again is only granted until the NV index representing the minimum required version exceeds this firmware's version:

$$Sig\big(K^{priv}_{Manu}, \big[TPM2\_PolicyPCR(Firmware_{v2}) \wedge TPM2\_PolicyNV(NV_{Version} \leq v2)\big]\big)$$

This leads to $Policy_{Seal}$ being:

$$\begin{aligned} Policy_{Seal} := & TPM2\_PolicyAuthorize\big(K^{pub}_{Manu}, \\ & \big[TPM2\_PolicyPCR(Firmware_{v1}) \wedge \\ & \quad TPM2\_PolicyNV(NV_{Version} \leq v1)\big] \\ & \vee \big[TPM2\_PolicyPCR(Firmware_{v2}) \wedge \\ & \quad TPM2\_PolicyNV(NV_{Version} \leq v2)\big]\big) \end{aligned}$$

During the update process, the updater mode stores the firmware including its policy on the device's flash drive. However, it does not require access to the encrypted container.

A similar concept with TPM 1.2 would have required that the updater mode would have required access to the encrypted container and to reseal the secret, or the manufacturer would have to have known the secret and bind it for the new PCR values. This is one of the main benefits of this TPM 2.0 based approach.

Firmware Runtime. Whenever a legitimate firmware version starts, it can unseal the necessary data and read/write to these storage areas. In order to invalidate access by outdated firmware versions, during each start, the firmware will check the currently stored minimal required firmware version inside the TPM counter and increment it to its own firmware version. This invalidates the usage of PolicyAuthorize branches for previous firmware version, since they require a lower value for the $NV_{Version}$ counter. Firmware should only perform this increment when it successfully completed its self test and started up correctly, since a recovery of the previous version is impossible afterwards. Instead the issuing of a new firmware version would be required. Of course a manufacturer may choose to issue a certain recovery firmware version, or multiple such version by, e.g., encoding those as the "odd version" vs. regular firmware as "even versions".

We write the increment of the NV counter as

$$TPM2\_NV\_Increment(NV_{Version}, v1)$$

which invalidates any older policies and thereby any outdated firmwares. This leads to:

$$\begin{aligned} Policy_{Seal} := & TPM2\_PolicyAuthorize\big(K^{pub}_{Manu}, \\ & \big[TPM2\_PolicyPCR(Firmware_{v1}) \wedge \\ & \quad TPM2\_PolicyNV(NV_{Version} \leq v1)\big]\big) \end{aligned}$$

A similar concept with TPM 1.2 could only have removed the sealed blobs for older versions from the flash drive, but in case of a readout of the flash from an earlier state, there would have been no possibility to actually disable these older policies with the TPM. This is another main benefit of this TPM 2.0 based approach.

3.4 Security Considerations

The presented concept relies on the correct cryptographic and functional execution of a TPM 2.0 implementation for the encryption and correct handling of the policy tickets respectively. Furthermore, the scheme relies upon a set of specific assumptions in order to function properly:

- The private ticket signing key of the vendor must be kept confidential.
- The provisioning needs to use the correct ticket public key for verification.
- The scheme does not protect against runtime-attacks against software.

3.5 Prototypical Implementation

The described concept was implemented in a proof-of-concept demonstrator for a typical automotive Head Unit. An Intel NUC D34010WYK in combination with Tizen In-Vehicle Infotainment (IVI) [14] using Linux kernel 4.0 was chosen for this implementation. These 4th generation NUCs are one of first commercial off-the-shelf (COTS) devices equipped with a TPM 2.0 implementation. Our demonstrator is shown in Fig. 1.

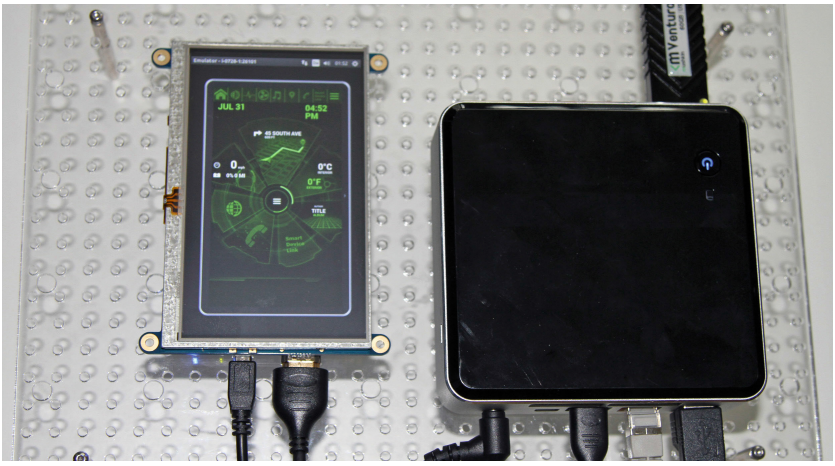


Fig. 1. TPM 2.0 head unit demonstrator

To protect IPR and privacy sensitive data, the Linux LUKS implementation (Linux Unified Key Setup) is used to create and to open an encrypted container for storing this data. The key for the encrypted container in turn is protected by TPM 2.0's enhanced authorization mechanisms as described in Sect. 2.6.

For ease of demonstration the representation of firmware versions in the PCR values was simplified – namely not calculations of the overall firmware hashes. Instead of measuring the complete firmware at boot and extending a PCR with this measurement, we measure a single file into the PCR called *version_file*. A TPM monotonic counter is used to represent the version number.

Note that the standard Linux Integrity Measurement Architecture (IMA) [15] could also not be used in practice for this scenario. Due to the event-driven startup mechanisms under modern Linux systems, the exact order of PCR extension can vary, which renders them unusable for sealing. Instead, the complete image would have to be measured from within the initial ramdisk. A future publication will demonstrate a possible approach based on e.g. [13].

In Algorithms 1 and 4, we assume that the TPM 2.0 equivalent of a Storage Root Key (SRK) – a TPM Primary Key under the Storage Hierarchy – is already computed. If the NV counter NVC is already defined, it is used directly. Otherwise NVC gets defined.

Provisioning. When the device is started for the first time, the following steps shown in Algorithm 1 are executed for provisioning the protected storage. It consists of the definition of the NV version counter, the instantiation of the policy, the creation of the LUKS container and the sealing of the LUKS key.

Algorithm 1 (Provisioning)

$/*\ NV\ Creation\ */$
if $not\ defined(NVC)$ ***then***
  $NVC := TPM2\_NV\_DefineSpace(counter)$
$/*\ Policy\ Initialization\ */$
$pubkey := TPM2\_LoadExternal(pub\_key\_manu)$
$policy := TPM2\_PolicyAuthorize(pubkey)$
$/*\ LUKS\ creation\ */$
$S := generate\_random\_key()$
$create\_crypto\_fs(S)$
$/*\ Sealing\ */$
$SRK := get\_srk()$
$enc_{SRK}(S) := TPM2\_Create(S, SRK, policy)$
$save(enc_{SRK}(S))$

To ease readability, we wrapped some of the details within simplified function calls. All functions prefixed with $TPM2\_$ are equivalent of corresponding TPM functions. $not\ defined(NVC)$ will check, whether the NV index has been defined by performing a $TPM2\_NV\_Read$ and checking the resulting value. The public key pub_key_manu of the manufacturer, loaded into the TPM, is bound to the object S encrypted by SRK via the policy calculateded by $TPM2\_PolicyAuthorize$. The corresponding private key to pub_key_manu now can be used to alter policies necessary for unsealing the encrypted key S.

Firmware Release. Algorithm 2 shows how the manufacturer can produce signatures for new firmware versions with a corresponding TPM 2.0 policy without using a TPM. The function $compute\_policy$ calculates the policy based on the PCR value, after extending a PCR by the firmware digest, and the version of the firmware. This computation is performed by the manufacturer according to the TPM2.0 specification. This policy will be signed with the private key of the

manufacturer. The device later must be able to associate the policy and the signature with the version to be installed.

Algorithm 2 (Firmware release)
$policy(version) := compute\_policy(digest(version\_file), version))$
$signature(version) := sign(policy(version), priv\_key\_manu)$

Upgrade. Algorithm 3 shows the steps executed to install a new firmware version signed by the manufacturer. The new version will be active after the next reboot (see Algorithm 4). The version of a signed firmware to be installed must be greater or equal to the current NV counter because this demonstrator should be able to execute the provisioning process several times and the hardware TPM's monotonic counter cannot be decremented again. To be more precise, even if the counter is undefined by $TPM2\_NV\_UndefineSpace$ it is not possible to reset the counter to a smaller value because for the next definition the counter will be initialized to be the largest count held by any NV counter over the lifetime of the TPM. In practice it would be possible to store the initial policy directly in the firmware image for version 1.

Algorithm 3 (Upgrade)
$version := get\_latest\_signed\_version\_number()$
$signature(version) := get\_signature(version)$
$policy(version) := get\_policy(version)$
$save(version\_file, version)$

In order to perform the simulated upgrade of the firmware binary, we increment the value encoded within the firmware representing version file.

Runtime. The steps described in Algorithm 4 are executed in the boot process to mount the encrypted container. They consist of the PCR extension with the digest of the version file, the satisfaction of the policy, using this policy for unsealing the container key, the opening of the encrypted container and then the invalidation of old firmware by incrementing the NV version counter.

Algorithm 4 (Mount encrypted file system)
/ Satisfying the policy */*
$version := load(version\_file)$
$TPM2\_PCR\_Extend(digest(version\_file))$
$approved\_policy := load\_policy(policy(version))$
$signature := load\_signature(signature(version))$
$pubkey := TPM2\_LoadExternal(pub\_key\_manu)$
$ticket := TPM2\_VerifySignature(signature, pubkey, approved\_policy)$
$session := TPM2\_StartPolicySession()$
$session.TPM2\_PolicyPCR(PCR)$
$session.TPM2\_PolicyNV(NVC, <=, version)$
$policy := TPM2\_PolicyAuthorize(pubkey, ticket, approved\_policy)$

```
/* Unsealing the container key and mount */
SRK := get_srk()
enc_SRK(S) := load()
TPM2_Load(enc_SRK(S))
S := TPM2_Unseal(enc_SRK(S), session)
mount_crpyto_fs(S)
firmware_self_test()
/* Invalidate old firmwares */
while TPM2_NV_Read(NVC) < version do
    TPM2_NV_Increment(NVC)
```

The functions *load_policy* and *load_signature* will load the policy respective the signature associated by the manufacturer with the version stored in the version file. The $session.TPM2\_Policy$ function represents an execution of those policy commands on the TPM, referring to the policy session *session*. For the signed approved policy, a ticket derived from this policy and the public key of the manufacturer is computed by $TPM2\_VerifySignature$, if the signature verification is successful. The current policy value of the session will be compared with the approved policy and the TPM then validates that the parameters to $TPM2\_PolicyAuthorize$ match the values used to generate the ticket. After the crypto file system is mounted, the device should perform a self test $firmware\_self\_test$ and the NV counter will be incremented until the value which corresponds to the current firmware version is reached. Thus, the object $enc_{SRK}(S)$ can't be unsealed by firmware versions less than *version*, providing protection against downgrade attacks.

4 Conclusion

In this paper, we discussed the benefits of integrating a TPM 2.0 (and the corresponding TSS 2.0) as HSM in an embedded system and presented a new concept for advanced remote firmware upgrades of such an embedded system while additionally enforcing intellectual property rights (IPR) and privacy protection for device-resident data. This new approach is only possible by using new features introduced by TPM 2.0 such as NV-RAM counters and Enhanced Authorization. Our approach secures device-resident data by ensuring that only new firmware upgrades of the manufacturer can be installed and downgrade attacks or attempts to install malicious firmware upgrades are prevented. In our prototypical implementation for an automotive head unit, we protected device-resident data of the manufacturer (i.e., navigation maps) and of the car user (i.e., contacts and preferred navigation destinations) against unauthorized access before, during, and after an upgrade. Our general concept can be applied in different application scenarios using different TPM and TSS 2.0 profiles.

References

1. How to Downgrade a PSP. http://www.wikihow.com/Downgrade-a-PSP
2. Android: Verified boot. https://source.android.com/security/verifiedboot/
3. Apple: iOS Security. Technical report Apple, p. 5 (2015)
4. ARM Security Technology: Security technology - building a secure system using trustzone technology (2009)
5. Challener, D., Arthur, W.: A Practical Guide to TpPM 2.0: Using the Trusted Platform Module in the New Age of Security. Apress, New York (2015)
6. Freescale: Secure Boot - For QorIQ Communications Processors (2011). http://cache.freescale.com/files/32bit/doc/white_paper/QORIQSECBOOTWP.pdf
7. Global Platforms: Trusted execution environment specifications (2011). http://www.globalplatform.org/specificationsdevice.asp
8. Greenberg, A., Miller, C., Valasek, C.: Hackers Remotely Kill a Jeep on the Highway - With Me in It. Wired, San Francisco (2015)
9. Henniger, O., Apvrille, L., Fuchs, A., Roudier, Y., Ruddle, A., Weyl, B.: Security requirements for automotive on-board networks. In: 9th International Conference on Intelligent Transport Systems Telecommunications (ITST 2009). IEEE (2009)
10. Hersteller Initiative Software (HIS) AK Security: SHE-Secure Hardware Extension, version 1.1st edn (2009)
11. Intel: Trusted Execution Technology
12. Intel: Software guard extensions programming reference (2014)
13. Corbet, J.: dm-verity. https://lwn.net/Articles/459420/
14. Linux Foundation, Tizen Association: Tizen. https://www.tizen.org/
15. Linux Weekly News: The Integrity Measurement Architecture. https://lwn.net/Articles/137306/
16. Miller, C., Valasek, C.: A Survey of Remote Automotive Attack Surfaces. Blackhat, Las Vegas (2014)
17. Trusted Computing Group: TPM Main Specification, Level 2 Version 1.2, Revision 116th edn, March 2011
18. Trusted Computing Group: Trusted Platform Module Library Specification, Family 2.0, Level 00, Revision 01.16th edn, October 2014
19. Trusted Computing Group: TSS Feature API Specification, Family 2.0, Level 00, Revision 00.12nd edn, November 2014
20. Trusted Computing Group: Algorithm Registry, revision 01.22nd edn (2015)
21. Trusted Computing Group: PC Client Platform TPM Profile (PTP) Specification, Family 2.0, Revision 00.43rd edn, January 2015
22. Trusted Computing Group: TCG TpPM 2.0 Library Profile for Automotive-Thin, version 1.0edn, March 2015
23. Trusted Computing Group: Trusted Computing Group TpPM 2.0 Library Specification Approved as an ISO/IEC International Standard, June 2015
24. Trusted Computing Group: TSS System Level API and TPM Command Transmission Interface Specification, Family 2.0, Revision 01.00edn, January 2015
25. Weyl, B., Wolf, M., Zweers, F., Gendrullis, T., Idrees, M.S., Roudier, Y., Schweppe, H., Platzdasch, H., El Khayari, R., Henniger, O., et al.: Secure on-board architecture specification. Evita Deliverable D3.2 3, 2 (2010)
26. Wilkins, R., Richardson, B.: UEFI Secure Boot in Modern Computer Security Solutions. Technical report, UEFI Forum (2013)
27. van Woudenberg, J.: 20 ways past secure boot. https://www.riscure.com/documents/10_ways_past_secure_boot_v1.0_jvw_shakacon.pdf

Sidechannel Analysis

RegRSA: Using Registers as Buffers to Resist Memory Disclosure Attacks

Yuan Zhao[1,2,3], Jingqiang Lin[1,2], Wuqiong Pan[1,2(✉)], Cong Xue[1,2,3], Fangyu Zheng[1,2,3], and Ziqiang Ma[1,2,3]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{yzhao,linjq,wqpan,cxue13,fyzheng,zqma13}@is.ac.cn
[2] Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China

Abstract. Memory disclosure attacks, such as cold-boot attacks and DMA attacks, allow attackers to access all memory contents, therefore introduce great threats to plaintext sensitive data in memory. Register-based and cache-based schemes have been used to implement RSA securely, at the expense of decreased performance. In this paper, we propose another concept named *register buffer*, which makes use of all available registers as secure data buffer, no matter scalar registers or vector registers. The plaintext sensitive data only appear in register buffer. Based on this concept, we finish a security implementation of 2048-bit RSA called *RegRSA*, to defeat against memory disclosure attacks. The 1024-bit Montgomery multiplication in RegRSA runs entirely in register buffer, by performing computations using scalar instructions and registers, maintaining intermediate variables in vector registers. Due to the size limitation of register buffer, several variables out of Montgomery multiplications are spilled into memory. RegRSA encrypts these variables with AES before saving in memory. Furthermore, RegRSA employs a windowing method and the CRT speed-up to accelerate RSA, and minimizes the data exchange between registers and memory to reduce the workload of AES encryption/decryption. The evaluation on Intel Haswell i7-4770R shows that, the performance of RegRSA achieves a factor of 0.74 compared to the regular RSA implementation in OpenSSL and is much greater than PRIME, the existing register-based scheme for 2048-bit RSA. Moreover, RegRSA allows multiple instances to run on a multi-core CPU simultaneously, which makes it more practical for the real-world applications.

Keywords: Memory disclosure attack · Register · RSA · Montgomery multiplication

Y.Zhao—This work was partially supported by National 973 Program under award No. 2014CB340603 and No. 2013CB338001, and Strategy Pilot Project of Chinese Academy of Sciences under award No. XDA06010702.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 293–307, 2016.
DOI: 10.1007/978-3-319-33630-5_20

1 Introduction

RSA [26] is the most prevalent asymmetric cryptographic algorithm. Although this algorithm is considered computationally secure, the RSA implementations face various security threats. Memory disclosure attacks, such as cold-boot attacks [15] and DMA attacks [28], allow attackers to obtain all memory contents, which makes plaintext private keys in memory unsafe. Besides, the sensitive data used or produced during RSA private-key operations, which can be used to derive the private key, should not appear in memory in plaintext. These sensitive data shall be stored in some secure storage when they participate in the private-key computations.

Registers and L1 caches in CPUs dedicated to one core, become the required secure storage because they are exclusive to the thread currently running on this CPU core under certain controls [10,11]. L1 caches are much larger than registers and programmes can be coded in high-level languages. But malicious binaries running on one CPU core could exploit the last-level cache (LLC) to flush a L1 cache line of other cores to memory with the hardware cache coherence mechanism. The existing cache-based scheme [11] forces all other cores that share LLC, into the no-fill cache mode during the cryptographic computations, and sharply reduces the memory access performance of these cores. Moreover, it does not support multiple instances on the cores that shares LLC. The advantage of register-based scheme is that registers are unaffected by other cores, which builds the possibility of executing multiple instances on a multi-core CPU simultaneously. However, the challenge is that registers might be not enough for the asymmetric cryptographic implementation, which requires (sensitive) data to be swapped between registers and memory frequently (results in frequent symmetric encryptions before being written into memory and decryptions after being loaded into registers). So the key point of register-based schemes is to implement the most frequent function entirely in registers as far as possible.

In order to implement an efficient register-based RSA system against memory disclosure attacks, we should choose an implementation method of RSA which has advantages both on speed and storage consuming. Redundant representation [14] is the major method for vector-instruction implementations, which achieves high speed but demands much more storage space. PRIME [10] adopts redundant representation method. In order to finish register-based RSA implementation, PRIME has to abandon CRT method and its performance is greatly degraded. Another vector approach [6] adopts 2-way single instructions to implement Montgomery multiplication, which consumes less storage than redundant representation method. But the authors in [6] point out that its speed is lower than 64-bit scalar implementation. So the register-based scheme adopting 64-bit scalar instructions is our mainly concerned.

1.1 Contributions

In this paper, we propose an 2048-bit RSA implementation named *RegRSA*, resistant to memory disclosure attacks. Our basic idea is to keep all plaintext

sensitive data only in registers when being used in RSA private-key computations. The performance of RegRSA is close to the implementation in OpenSSL and it allows multiple instances to run on a multi-core CPU simultaneously. Our major contributions are described as follows:

- We propose a concept named *register buffer* that makes use of all available registers as secure data buffer. We also summarize the available registers in newer CPUs and the instructions used to move data between different kinds of registers.
- We make full use of 704-byte register buffer in Intel Haswell CPU to implement an 1024-bit Montgomery multiplication running entirely in register buffer, by performing computations using scalar instructions and registers, maintaining intermediate variables in vector registers.
- We use a fixed windowing method to speed up Montgomery exponentiation, and finish a CRT-enabled 2048-bit RSA implementation. The precomputed table and intermediate variables are encrypted and stored in OS kernel heap or stack. We present several improvements on using AES-NI [13] to reduce the cost of AES key expansion for each data block.

1.2 Related Work

Cold-boot attacks and DMA attacks have been explored to access sensitive data in memory. The first attack exploiting the remanence effect of RAM was reported in [1]. In 2008, the work in [15] presented an cold-boot attack which recovered cryptographic keys by freezing the RAM chips. The study in [28] provided an overview of cold-boot attacks and the proposed counter-measures. DMA attacks are launched from peripherals through high-speed peripheral ports like PCI [8] and Firewire [4]. TRESOR-HUNT [3] presented an advanced DMA attack to get the AES key in privileged registers by compromising the integrity of OS kernel.

In order to resist memory disclosure attacks, register-based and cache-based schemes are proposed. The register-based schemes employ registers as the secure storage, such as AESSE [23], Amnesia [27] and TRESOR [24] which keep AES keys in registers and computed AES using registers only. PRIME [10] is the first register-based scheme for 2048-bit RSA private-key operations, but its performance is greatly degraded. The study in [29] proposed an elliptic curve cryptography implementation using CPU registers. On the other hand, the cache-based schemes employ caches to store sensitive data. FrozenCache [25] exploited CPU caches to store keys outside RAM. Copker [11] proposed a method to perform RSA private-key operations within CPU caches. Existing RSA implementations against memory disclosure attacks including PRIME and Copker, cannot support simultaneous multiple instances on multi-core CPUs well. Mimosa [12] is the first work to protect sensitive data using hardware transactional memory, which essentially stores data in caches; but it requires special CPU hardware features.

1.3 Outline

The rest of this paper is organized as follows. Section 2 introduces the available registers in CPU. Sections 3 and 4 describe the design and the implementation of RegRSA. In Sect. 5, we evaluate RegRSA in terms of security and performance. We conclude this paper in Sect. 6.

2 Available Registers in Commodity CPUs

Registers are classified into user-accessible registers and special internal registers. User-accessible registers can be read or written by CPU instructions, while internal registers cannot be accessible by instructions. On Intel CPUs, the most commonly-used x86 platform, instructions include scalar instructions and vector instructions. Firstly, scalar integer instructions operate on scalar registers, also called general purpose registers (GPRs); and scalar floating-point instructions operate on floating-point registers. Secondly, Intel vector instruction sets include MMX [9], SSE [18] and AVX [21]. The registers for MMX are called MM registers, which are physically the same registers with floating-point registers. The registers for SSE are called XMM registers, and the registers for AVX are the extensions of XMM registers called YMM registers. XMM registers are the low 128-bit of YMM registers. A 64-bit CPU has more registers and every scalar register is double size. Users on 64-bit OS can manipulate these greater-size registers. For example, on an Intel Haswell CPU, there are sixteen 64-bit GPRs, eight 64-bit MM registers and sixteen 256-bit YMM registers. The total space of these registers is 704-byte. Besides, there are four 64-bit debug registers (DRs) available if the operating system prohibits debugged applications access these registers [24].

Table 1. Instructions used to move data between different kinds of registers

	GPR	MM	XMM	YMM	DR
GPR	MOV	MOV	VMOV/PINR/VPINR	-	MOV
MM	MOV	MOV	MOVQ2DQ	-	-
XMM	VMOV/PEXTR/VPEXTR	MOVDQ2Q	MOVDQA/VMOVDQA	VINSERTI128	-
YMM	-	-	VEXTRACTI128	VMOVDQA	-
DR	MOV	-	-	-	-

Scalar registers, MM registers and YMM registers can be used as secure storage for sensitive computations. The data in different kinds of registers may be exchanged frequently. Table 1 summarizes the instructions used to move data between different kinds of registers. Note that, scalar registers and XMM registers can exchange data with most other registers, so they could be the hub of data exchange or keep the most commonly used data. Specifically, instructions PINR and VPINR insert a 64-bit data item from a scalar register to the particular location in a XMM register. The difference between PINR and VPINR is

that, PINR is a legacy instruction that keeps the high 128-bit of the destination YMM register unchanged, but with performance penalty. VPINR is an new AVX instruction which will clear the high 128-bit of destination YMM register with no performance penalty.

3 System Design

In this section, we present the design goals of RegRSA. Then, we propose the concept of register buffer and describe the architecture of RegRSA on top of register buffer.

3.1 Design Goals

The target of this work is to design a *secure* and *efficient* 2048-bit RSA implementation as follows.

Security Goal. All the sensitive data including cryptographic keys (AES keys and RSA private keys) and intermediate variables does not appear in the memory in the form of plaintext, against cold-boot attacks [15] and other memory disclosure attacks.

Performance Goal. The speed of RegRSA should be close to regular implementations, e.g., OpenSSL. Multiple optimization techniques are expected to be exploited in RegRSA, such as Montgomery multiplication [22], the windowing method [19] for modular exponentiation, and the CRT speed-up [19].

Assumptions. First, the OS kernel is trustworthy, which means an attacker can not tamper the OS kernel to launch attacks such as TRESOR-HUNT [3]. Second, the system initialization before any user-space process is safe for users to input a password to derive an AES key in privileged debug registers [24]. Finally, the register features are available in hardware and software platform, that is, CPUs and the OS are 64-bit, and necessary instruction extensions including AVX, AES-NI and MULX are ready.

3.2 Register Buffer

Existing register-based schemes [10,24,29] have investigated the usage of registers for storing sensitive data, but only focused on some registers (not all available registers). The high-speed implementations of cryptographic algorithms have explored the cooperation of different kinds of instructions and registers to accelerate cryptograph computing, but such implementations do not systematically consider the security of keys.

In this paper, we propose a concept named register buffer, which makes use of all available registers as secure data buffer, no matter scalar registers or vector registers. As the registers are used to provide operands and accept results for certain instructions, register buffer requires the comprehensive cooperation of different kinds of instructions and registers for efficient computing and secure storage. As described in Fig. 1, register buffer is divided into two sets of registers:

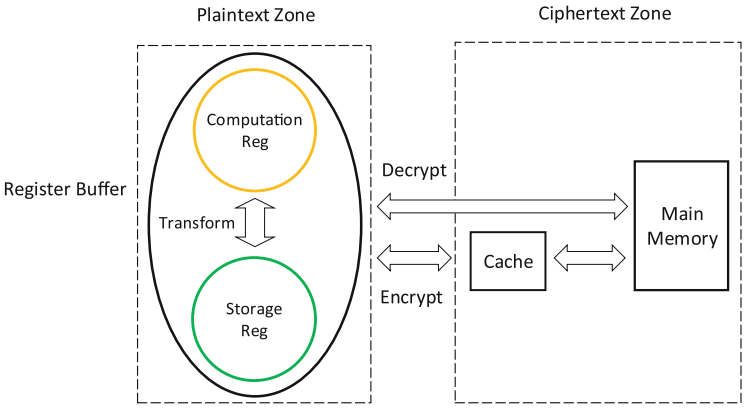


Fig. 1. Register buffer

computation-reg and *storage-reg*. Computation-reg is a set of registers which provides inputs and receives results for on-going instructions. Storage-reg is a set of registers for maintaining data unused in the current period. The registers are ready to be converted between computation-reg and storage-reg depending on which kind of instructions are being executed. Data are plaintext in register buffer. When data have to be stored in memory (RAM or caches), they are encrypted with AES. In brief, register buffer deems that all available registers are secure storage resources which should be fully utilized for efficiency and security.

3.3 RegRSA Architecture

From an implementation point of view, RegRSA is divided into three levels: (1) the modular multiplication level, (2) the modular exponentiation level and (3) the RSA level. The high level calls the lower level by sending parameters and receiving results. The architecture and the data transfer between registers and main memory are depicted in Fig. 2.

In the modular multiplication level, we design and implement an all-register 1024-bit Montgomery multiplication which computes Montgomery multiplication by using scalar instructions and registers, maintaining parameters in vector registers. In the modular exponentiation level, we apply the windowing method for 1024-bit Montgomery exponentiation. We compute, encrypt and save precomputed table in the kernel heap, and then load the precomputed values depending on the exponents and decrypt the values. Based on the CRT method, we implement 2048-bit RSA by performing two 1024-bit Montgomery exponentiations. The encrypted Montgomery exponents are loaded from memory and the results of Montgomery exponentiations are encrypted and swapped between registers and the kernel stack. In all levels, all the sensitive data in RAM are encrypted with AES and the AES key is in debug registers. Note that, the major computations,

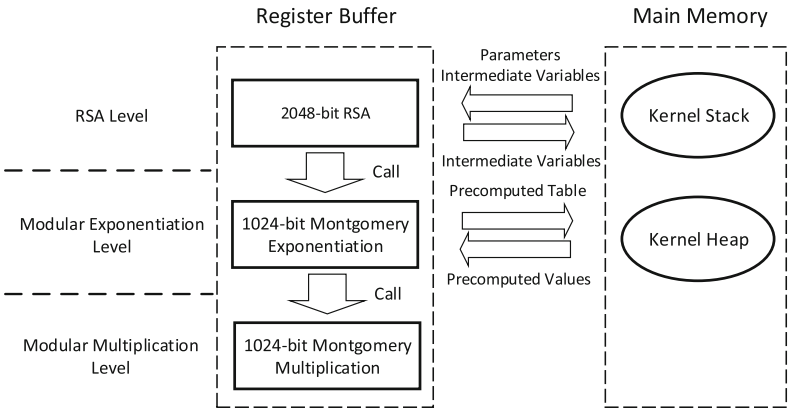


Fig. 2. RegRSA architecture

Montgomery multiplication, do not need to exchange data between register and memory, so the performance degradation from data encryption/decryption and exchange is not significant.

4 Implementation

In this section, we describe the detailed implementation of RegRSA: in particular, the assembly codes of RegRSA to gain the complete control of registers, and the integration of RegRSA into Linux kernel because such implementation must run in kernel mode otherwise task switching may dump registers into RAM.

4.1 Montgomery Multiplication Implementation

1024-bit Montgomery multiplication [22] performs the computation $S = A \times B \times R^{-1} \pmod{M}$, $R = 2^{1024}$, $0 \leqslant A, B < M < R$. Coarsely Integrated Operand Scanning (CIOS) [20] is an interleaved Montgomery multiplication method with three 1024-bit inputs A, B, M and one 64-bit input μ $(-M^{-1} \bmod 2^{64})$. As depicted in Fig. 3, we employ scalar registers and scalar instructions to perform Montgomery multiplication, while keeping three 1024-bit inputs in YMM registers, 64-bit input in a scalar register and saving the 64-bit intermediate variables q_j $(q_j = S_j \times \mu \bmod 2^{64})$ in MM registers. We make full use of 704-byte register buffer to finish the first all-register 1024-bit Montgomery multiplication implementation.

According to 64-bit Linux call convention, registers RBX, RBP, RSP, R12, R13, R14, R15 must be protected, we push these registers except RSP into stack. Since the left fifteen 64-bit scalar registers are not enough for computing the whole 1024-bit Montgomery multiplication in one time, we split Montgomery multiplication into four parts. The first part performs $A[0 \sim 7] \times B[0 \sim 7] + M[0 \sim 7] \times (q_0 \sim q_7)$, the second part performs $A[8 \sim 15] \times B[0 \sim 7] + M[8 \sim 15] \times$

$(q_0 \sim q_7)$, the third part performs $A[0 \sim 7] \times B[8 \sim 15] + M[0 \sim 7] \times (q_8 \sim q_{15})$ and the fourth part is $A[8 \sim 15] \times B[8 \sim 15] + M[8 \sim 15] \times (q_8 \sim q_{15})$. The j-th round computation of the first part is depicted in Fig. 3. Instruction MULX [18] is used to perform 64-bit scalar multiplication. The summation variable S occupies nine scalar registers and is updated in every round, and q_j in MM registers is moved back when needed. Eight MM registers are enough for storing $q_0 \sim q_7$ or $q_8 \sim q_{15}$ for eight rounds. Besides, the final subtraction in Montgomery multiplication is always performed, whether or not S is no less than M, to eliminate the timing side-channel [7].

4.2 Montgomery Exponentiation Implementation

We use the fixed windowing method [19] to speed up Montgomery exponentiation. The size of window is 6-bit (64 entries). For CRT-enabled RSA, two different moduli are needed; i.e., for the private-key parameters p and q, two tables of ${C_p}^k$ and ${C_q}^k$ $(k = 0, \cdots, 63)$ which share a memory space. At the beginning of Montgomery exponentiation, we prepare precomputed table: compute and encrypt ${C_p}^2 \sim {C_p}^{63}$ (or ${C_q}^2 \sim {C_q}^{63}$), save them into kernel memory. ${C_p}^0$, ${C_p}^1$ (or ${C_q}^0$, ${C_q}^1$) are also encrypted and saved in the precomputed table for the const time of table lookup. Then we get the precomputed value from the precomputed table to YMM registers depending on the exponent, and decrypt them using 128-bit AES. Since the maximum size of kernel stack is 8KB which still has to save struct `thread_info` at the stack bottom, 6-bit precomputed table which needs 8 KB memory cannot be saved in kernel stack. So we use system function `kmalloc` to allocate 8KB memory on the kernel heap for precomputed table before beginning RegRSA, and use function `kfree` to free memory after

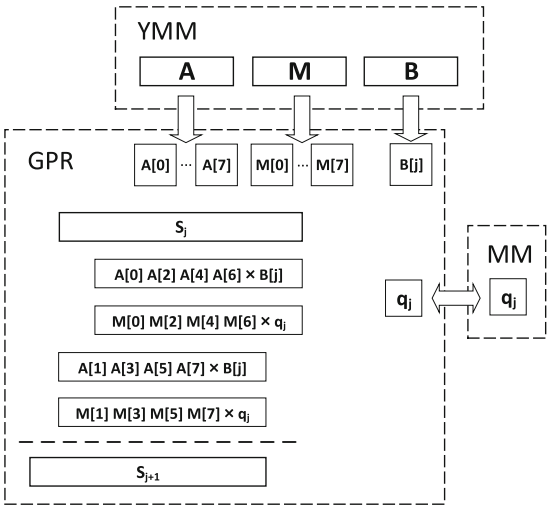


Fig. 3. First part of our 1024-bit Montgomery multiplication implementation

finishing RegRSA. We load all entries of the precomputed table in sequence when we need certain precomputed values.

AES-NI instruction extension [13] is used to implement AES encryption/decryption and key expansion by hardware. In this study, we present several improvements on using AES-NI. First, AES-128 and AES-256 only need one 128-bit temporary register in the key expansion. Second, we derive the round keys from the last round to the first round which is very useful for AES decryption. Third, we perform on-the-fly bulk AES encryption/decryption which uses one round key to process multiple 128-bit data blocks and then the next round key, which sharply reduces the cost of key expansions for each data block.

4.3 RSA Implementation

We utilize the CRT method and our 1024-bit Montgomery exponentiation implementation to finish 2048-bit RSA private-key operations. The input parameters are copied from the user space memory to the kernel space memory, including the CRT parameters p, C_p, d_p, q, C_q, d_q, $q^{-1} \bmod p$, and Montgomery parameters $R^2 \bmod p$, $R^2 \bmod q$, $-p^{-1} \bmod 2^r$, $-q^{-1} \bmod 2^r$. The parameters d_p, d_q, $q^{-1} \bmod p$, $R^2 \bmod p$, $R^2 \bmod q$, $-p^{-1} \bmod 2^r$, $-q^{-1} \bmod 2^r$ are constant for each private key, while $C_p = C \bmod p$ and $C_q = C \bmod q$ which need to be computed on the fly for each ciphertext C. All the above parameters are encrypted with AES when they are stored in memory. As described in Algorithm 1, the CRT speed-up requires two 1024-bit Montgomery exponentiations for 2048-bit RSA. As register buffer is not enough to hold the result of one Montgomery exponentiation while performing another, we keep the intermediate variables encrypted in memory.

Algorithm 1. Implementation of 2048-bit RSA Private-key Operation

Input: The CRT parameters p, C_p, d_p, q, C_q, d_q, $q^{-1} \bmod p$, and Montgomery parameters $R^2 \bmod p$, $R^2 \bmod q$, $-p^{-1} \bmod 2^r$, $-q^{-1} \bmod 2^r$

Output: Plaintext M.

1: Load parameters p, C_p, d_p, $R^2 \bmod p$, $-p^{-1} \bmod 2^r$ from RAM to registers and decrypt them with AES
2: $M_p \leftarrow {C_p}^{d_p} \mod p$
3: Encrypt M_p with AES and save it in RAM
4: Load parameters q, C_q, d_q, $R^2 \bmod q$, $-q^{-1} \bmod 2^r$ from RAM to registers and decrypt them with AES
5: $M_q \leftarrow {C_q}^{d_q} \mod q$
6: Encrypt M_q with AES and save it in RAM
7: Load M_p, M_q, q, $q^{-1} \bmod p$, $R^2 \bmod p$, $-p^{-1} \bmod 2^r$ from RAM to registers and decrypt them with AES
8: $M \leftarrow M_q + [(M_p - M_q) \times (q^{-1} \bmod p) \mod p] \times q$
9: **return** M

4.4 Integration in Linux Kernel

We integrate RegRSA into Linux kernel to ensure no data in registers would leak into main memory.

Char module. We integrate RegRSA into a char module and compile this module into Linux kernel. The module provides an interface for user space with the system call ioctl. The user processes can use the interface to send the inputs to RegRSA and receive the results. In kernel space, RegRSA can access privileged debug registers for AES keys (and all other registers).

Atomicity. Before the execution of RegRSA, kernel preemption is suspended by calling `preempt_disable` and interrupts are disabled by calling `local_irq_save`. So data in registers will not be written into main memory by context switch during the RegRSA computations. Finally, kernel preemption is restored by calling `preempt_enable` and interrupts are enabled by calling `local_irq_restore`. As non-maskable interrupts (NMIs) cannot be disabled by software settings, we adopt the solution in Copker [11]. We modify NMI handlers to clear registers with sensitive data, including scalar registers, MM registers and YMM registers.

4.5 AES Key

AES key is securely produced and maintains safety after OS initialization.

AES key derivation. By utilizing an existing technique in TRESOR [24], we have patched the linux kernel to let user input a password before any userland process startup. We assume the password is strong enough to defeat against brute-force attacks. Moreover, the user can update AES keys by changing the password after a while.

AES key protection. Also like in TRESOR [24], AES key is stored in debug registers which cannot be accessed from user space. System functions `ptrace_set_debugreg` and `trace_get_debugreg` are patched to ensure the user process cannot set four debug registers dr0 to dr3 or get their values.

5 Evaluation

In this section, we conduct security analysis on RegRSA, evaluate its performance and the impact of RegRSA. In the end, we discuss some further considerations. The system is evaluated on this platform: Intel Haswell i7-4770R CPU, 8GB memory and OS is Ubuntu 14.04 64-bit. We turn off Turbo Boost of Intel Haswell i7-4770R for stable frequency 3.2GHz, in the performance experiments.

5.1 Security Analysis

Memory Disclosure Attacks. First, we explore the security for the situation of only one RegRSA running instance on CPU. The input parameters

are all sensitive data, including CRT parameters and Montgomery parameters, which are encrypted with AES before passing from user space to kernel space. AES key is in privileged debug registers, no user process could read or write these registers. Due to the atomic execution of RegRSA, no data in registers will leak into main memory by task switch. In the Montgomery multiplication level, all the intermediate variables are produced and kept in registers. In the Montgomery exponentiation level, the precomputed table is encrypted before stored in RAM, and the precomputed values are decrypted in XMM registers. In the RSA level, the results of Montgomery exponentiation are encrypted before stored in memory and decrypted after reading into registers. All the data in registers are eliminated before leaving the atomic region. In a word, all the sensitive data in main memory are encrypted, and the plaintext sensitive data appear in registers only.

Also, we have done experimental evaluation on RegRSA. We used Kdump to dump kernel memory while RegRSA was performing RSA private-key operations. We searched AES keys and RSA private keys in the captured image and found no matching strings.

When there are multiple instances of RegRSA on several CPU cores, no matter how many requests received from user space, only one RegRSA instance is running on a single CPU core in a moment due to the atomic execution. As each RegRSA instance owns its own variables in kernel stack and heap, RegRSA can execute multiple instances on different CPU cores which will not interfere with each other.

Cache-based Timing Side-Channels. RegRSA is resistant to cache-based timing side channel attacks [2,5,7]. We employ the fixed windowing method in Montgomery exponentiation and the final subtraction in Montgomery multiplication is always performed, so there is no branch in the execution flow. Thus, there is no timing side channels in RegRSA based on instruction paths. When we perform the table-lookup in Montgomery exponentiation, we load the precomputed table as a whole. This makes attackers could not learn which entry RegRSA accesses and deduce the exponents. Therefore, there is no timing side channels attacks based on data access.

5.2 Performance

Comparison with OpenSSL. We launch different numbers of threads in user space to send RSA private-key operation requests to RegRSA. Because of simultaneous multithreading (SMT), eight threads make RegRSA occupy the CPU fully. RegRSA is compared with OpenSSL version 1.0.1f, in different concurrent levels. The numbers of RSA private-key operations per second and the ratios of the performances between RegRSA and OpenSSL are given in Table 2.

As the number of threads increases, the performance becomes better, either for RegRSA or OpenSSL. RegRSA achieves at least a factor of 0.74 compared to the speed of OpenSSL. The efficiency degradation of 26 % is acceptable, to defeat against memory disclosure attacks.

Table 2. Comparison with OpenSSL

# of Concurrent Threads	1	4	8
RegRSA	637	2537	2638
OpenSSL	858	3308	3571
RegRSA / OpenSSL	0.74	0.77	0.74

Comparison with PRIME. We expect to compare RegRSA with PRIME [10] on the same platform. Because we do not have the source code of PRIME, the comparison with PRIME is conducted through OpenSSL – we assume the OpenSSL in [10] is identical with that in this paper (version 1.0.1f). Table 3 presents the speed ratio of PRIME and RegRSA to OpenSSL, respectively. So we can see that the efficiency of RegRSA is far beyond PRIME; moreover, RegRSA can execute 8 instances on quad-core CPUs simultaneously.

5.3 Impact on Concurrent Tasks

As RegRSA disables kernel preemption and interrupts during its running, the stable of the operating system and the performance of other applications may be effected. We initiate eight user threads to continuously send RSA private-key operation requests. Through a period of observation, we see that the operating system works properly without disruptions, and the response of OS is normal as well. Then we use the SysBench benchmark to evaluate the performance impact on concurrent tasks. We run the CPU test of SysBench and execute the test in four situations. The first is no computing-intensive tasks performing during the test. The second is same as OS stable test with running eight user threads to send requests to RegRSA. The third is to start eight threads to perform a "mock" RegRSA in user space. This mock RegRSA does not disable kernel preemption and interrupts, and use a fixed value as the AES key – other configurations are the same as those of RegRSA. The fourth is to perform OpenSSL speed test for 2048-bit RSA private-key operations in eight threads. Test parameters of SysBench are 8 threads, 10,000 requests and prime numbers up to 20000. The test score is the average time for each request.

The results are presented in Table 4. There is no significant difference between the situations of RegRSA, mock RegRSA and OpenSSL. So disabling kernel preemption and interrupts in RegRSA does not cause obvious negative effects.

Table 3. Comparison with PRIME

	Latency (ms)	Speed Ratio
PRIME	21.0	-
OpenSSL [10]	1.8	-
PRIME / OpenSSL [10]	-	0.086
RegRSA / OpenSSL 1.0.1f	-	0.74

Table 4. Impact on other applications

	Idle	RegRSA kernel space	Mock RegRSA user space	OpenSSL
SysBench (ms)	2.83	5.94	5.87	5.98

5.4 Discussions

SMT. SMT on Intel CPUs also known as Hyper-Threading (HT), which is used to improve the efficiency of processing units in CPUs. HT provides two hardware threads on each core. Each thread has a separate set of registers that can be used for RegRSA. So RegRSA can run at most eight instances on a quad-core HT CPU simultaneously. Two threads on one CPU core will facilitate instruction pipelining which improves performance of RegRSA slightly; see Table 2 for details.

Full Memory Encryption. The aim of full memory encryption is to provide confidentiality of the entire software stack outside the CPU [17]; therefore, memory disclosure attacks are defeated. However, existing memory encryption suffer significant performance degradation [16].

Other CPUs. If a CPU possesses multiple registers including scalar registers, MM registers and YMM registers, and supports AES-NI instruction set and MULX instruction, it may be a suitable platform for RegRSA. So besides Intel CPUs, other CPUs like AMD can also be considered. For example, AMD Carrizo CPUs also support AVX2, MULX and AES-NI, it can be a potential candidate.

6 Conclusion

We propose a concept named register buffer that makes use of all available registers as secure data buffer, and design and implement 2048-bit RSA named RegRSA. In RegRSA, all the sensitive data appeared in main memory are encrypted, and the plaintext data are protected in registers to defeat against memory disclosure attacks. The evaluation on Intel Haswell i7-4770R showed that, the performance of RegRSA achieves at least a factor of 0.74 compared to the RSA implementation of OpenSSL. Moreover, RegRSA supports multiple instances on multi-core CPUs simultaneously, which makes RegRSA more practical for the real-world applications against memory disclosure attacks. We will explore to use two sets of registers of Hyper-Threading for one RSA computation instance in the future.

References

1. Anderson, R., Kuhn, M.: Tamper resistance - a cautionary note. In: 2nd Usenix Workshop on Electronic Commerce, vol. 2, pp. 1–11 (1996)
2. Bernstein, D.: Cache-timing attacks on AES (2005)

3. Blass, E.-O., Robertson, W.: TRESOR-HUNT: Attacking CPU-bound encryption. In: 28th Annual Computer Security Applications Conference, pp. 71–78. ACM (2012)
4. Böck, B., Austria, S.B.: Firewire-based physical security attacks on windows 7, efs and bitlocker. Secure Business Austria Research Lab (2009)
5. Bonneau, J., Mironov, I.: Cache-collision timing attacks against AES. In: 8th Workshop on Cryptographic Hardware and Embedded Systems, pp. 201–215 (2006)
6. Bos, J.W., Montgomery, P.L., Shumow, D., Zaverucha, G.M.: Montgomery multiplication using vector instructions. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 471–490. Springer, Heidelberg (2014)
7. Brumley, D., Boneh, D.: Remote timing attacks are practical. Comput. Netw. **48**(5), 701–716 (2005)
8. Carrier, B.D., Grand, J.: A hardware-based memory acquisition procedure for digital investigations. Digit. Invest. **1**(1), 50–60 (2004)
9. DuLong, C., Gutman, M., Julier, M., et al.: Complete Guide to MMX Technology. McGraw-Hill Professional, New York (1997)
10. Garmany, B., Müller, T.: PRIME: Private RSA infrastructure for memory-less encryption. In: Proceedings of the 29th Annual Computer Security Applications Conference, pp. 149–158. ACM (2013)
11. Guan, L., Lin, J., Luo, B., Jing, J.: Copker: Computing with private keys without ram. In: 21st ISOC Network and Distributed System Security Symposium (NDSS) (2014)
12. Guan, L., Lin, J., Luo, B., Jing, J., Wang, J.: Protecting private keys against memory disclosure attacks using hardware transactional memory. In: 36th IEEE Symposium on Security and Privacy (Oakland) (2015)
13. Gueron, S.: Intel Advanced Encryption Standard (AES) New Instructions Set (2010)
14. Gueron, S., Krasnov, V.: Software implementation of modular exponentiation, using advanced vector instructions architectures. In: Özbudak, F., Rodríguez-Henríquez, F. (eds.) WAIFI 2012. LNCS, vol. 7369, pp. 119–135. Springer, Heidelberg (2012)
15. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold-boot attacks on encryption keys. Commun. ACM **52**(5), 91–98 (2009)
16. Henson, M., Taylor, S.: Beyond full disk encryption: Protection on security-enhanced commodity processors. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 307–321. Springer, Heidelberg (2013)
17. Henson, M., Taylor, S.: Memory encryption: A survey of existing techniques. ACM Comput. Surv. (CSUR) **46**(4), 53 (2014)
18. Intel. Intel 64 and ia-32 architectures software developer's manual volume 2 (2a, 2b & 2c): Instruction set reference, a-z (2015)
19. Koc, C.K.: High-speed RSA implementation. Technical report, RSA Laboratories (1994)
20. Koç, Ç.K., Acar, T., Kaliski, B.S.: Analyzing and comparing montgomery multiplication algorithms. Micro, IEEE **16**(3), 26–33 (1996)
21. Lomont, C.: Introduction to intel advanced vector extensions. Intel White Paper (2011)
22. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **44**(170), 519–521 (1985)

23. Müller, T., Dewald, A., Freiling, F.C.: AESSE: A cold-boot resistant implementation of AES. In: 3rd European Workshop on System Security, pp. 42–47. ACM (2010)
24. Müller, T., Freiling, F.C., Dewald, A.: TRESOR runs encryption securely outside RAM. In: USENIX Security Symposium, p. 17 (2011)
25. Pabel, J.: Frozen cache. Blog (2009). http://frozenchache.blogspot.com
26. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
27. Simmons, P.: Security through amnesia: a software-based solution to the cold boot attack on disk encryption. In: 27th Annual Computer Security Applications Conference, pp. 73–82. ACM (2011)
28. Wetzels, J.: Hidden in snow, revealed in thaw: Cold boot attacks revisited. arXiv preprint (2014). arXiv:1408.0725
29. Yang, Y., Guan, Z., Liu, Z., Chen, Z.: Protecting elliptic curve cryptography against memory disclosure attacks. In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) ICICS 2015. LNCS, vol. 8958, pp. 49–60. Springer, Heidelberg (2015)

Uncertain? No, It's Very Certain!

Recovering the Key from Guessing Entropy Enhanced CPA

Changhai Ou[1,2], Zhu Wang[1(✉)], Degang Sun[1], Xinping Zhou[1,2], and Juan Ai[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{ouchanghai,wangzhu,sundegang,zhouxinping,aijuan}@iie.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. It has always been the concern of side channel analysis that how to recover the key with a probability of about 1.00 under the condition that the number of power traces is very small and the success rates is very low. In order to recover the key, the attacker has to try to reduce the guessing entropy to decrease the uncertainty of the key. Unfortunately, guessing entropy is only a evaluation of attack ability in most cases. In this paper, we introduce the statistical characteristics of guessing entropy and propose guessing entropy enhanced CPA (GE-CPA). Its feasibility is verified in theory and experiment. Experiments on both AES algorithm implemented on an AT89S52 single chip and power trace set *secmatv*1 of DES encryption on the side channel attack standard evaluation board(SASEBO) from the website *DPA contest v*1. The experimental results show that, by only repeating the experiments less than 30 times, our GE-CPA can effectively recover the key even under the bad condition that success rate only ranges from 5 % to 8 %. Thus, the problem is well solved.

Keywords: Guessing entropy · CPA · Guessing entropy enhanced CPA · GE-CPA · Side channel · *DPA contest v*1

1 Introduction

Standaert et al. proposed two evaluation methods success rate and guessing entropy to evaluate the efficiency of side channel attacks [14]. Souissi et al. detailed that, on one hand, the first-order success rate denoted the probability that, given a pool of traces, the attack's best guess was the correct key; On the other hand, the guessing entropy measured the position of the correct key in a list of key hypotheses ranked by a kind of side channel attack [13]. A. Venelli expressed guessing entropy as the average position of the correct hypothesis in the sorted hypothesis vector of an attack [16]. The positions of wrong keys are not taken into consideration.

In short, as well as the concept of entropy firstly proposed by Shannon [12], which denotes the uncertainty of information, guessing entropy is also used to

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 308–320, 2016.
DOI: 10.1007/978-3-319-33630-5_21

denote the uncertainty of information. The higher the guessing entropy, the greater uncertainty of the key.

1.1 Related Works

There are three main types of applications of the guessing entropy:

Firstly, as well as success rate in [6,11,15,17], guessing entropy is used to evaluate the efficiency of side channel attacks. In other words, guessing entropy is used to evaluate the uncertainty of key [4,7,10].

Secondly, as well as entropy, guessing entropy is also used to evaluate the uncertainty of information. The information here is not limited to the key. For example, Michael Backes and Boris Köpf proposed a novel approach for quantifying a system's resistance to unknown message side channel attacks using guessing entropy [2].

Thirdly, guessing entropy is used to improve the efficiency of side channel attacks. Nassar et al. proposed an empirical approach named Rank Corrector (RC) aiming at enhancing most side channel attacks [9]. The main principle of *RC* is to detect and discard the false keys hypotheses when analyzing the ranking evolution. With the increase number of power traces, the correct key will reach the first position. Martin et al. constructed an extremely efficient algorithm that accurately computing the rank of a (known) key in the list of all keys [8]. This approach is tweaked and can be also utilised to enumerate the most likely keys in a parallel fashion.

However, the above improvements only consider that guessing entropy changes with the number of power traces used in each repetition. They don't consider the relationship between the repetitions and guessing entropy. So, very different from [8], we make use of the information leaking from the guessing entropy to recover the key rather than only a kind of evaluation in this paper. Our scheme significantly improves the efficiency of CPA.

1.2 Our Contributions

It is difficult to recover the key using in cryptographic device with a probability of about 1.00 under the condition that the number of power traces is very small and the success rate is very low. To solve the problem, we propose guessing entropy enhanced CPA (GE-CPA) in this paper. Its feasibility is proved in theory and experiment. Experiments on an AT89S52 single chip and the Side Channel Attack Standard Evaluation Board (SASEBO) show that, our scheme can effectively recover the key with a probability of about 1.00 even under the condition that the success rate of traditional CPA is only about 5 % to 8 %. Thus, the problem is well solved.

1.3 Organization

This paper is organized as follows. The concept of guessing entropy in side channel attack is given in Sect. 2. In Sect. 3, we introduce our guessing entropy

enhanced CPA (GE-CPA). Then, in Sect. 4, experiments on AES algorithm implemented on an AT89S52 single chip and power trace set *secmatv*1 on DES algorithm implemented on the side channel attack standard evaluation board (SASEBO) from the website *DPA contest V*1 [1] are performed to compare our GE-CPA with traditional CPA. Finally, we conclude this paper in Sect. 5.

2 Guessing Entropy

Similar to Shannon entropy, guessing entropy (GE) [14] in side channel attacks also denotes the uncertainty of information. Guessing entropy is defined as the key position in side channel attack. For example, the attacker performs CPA on AES and gets 256 correlation coefficients corresponding to guessing key from 0 to 255. Then, the correlation coefficients are sorted in descending order. The bit length of key corresponding to the same intermediate value is 8. For each 2^8 possible key in each experiment, if the guessing entropy of guessing key $k^{'}\left(0 \leq k^{'} \leq 2^8 - 1\right)$ is in the position ν, the guessing entropy is equal to ν.

The guessing entropy, which represents the uncertainly of the key used in cryptographic device, is widely used to evaluate the efficiency of side channel attacks.

3 Guessing Entropies Enhanced CPA

In this section ,we will introduce the statistical characteristics of guessing entropy. Then, we introduce the way to determine the correct key from guessing entropy enhanced CPA (GE-CPA). The attack flow and the special success rate of our GE-CPA are also given in this section.

3.1 The Statistical Characteristic of Guessing Entropy

We have shown the guessing entropy defined by Standaert et al. [14] in Sect. 2. The location i of the correct key s is returned for each experiment. In this paper, We assume that all keys are likely to be the correct one. However, the correct key is the most superior one of all possible keys. Thus, different to [14], in order to select the optimal key, we calculate guessing entropies for all possible keys. Each guessing key returns a guessing entropy for each repetition. We sort the guessing entropies in descending order and assign different scores according to their positions in the guessing vector. For example, small guessing entropies are given higher scores.

Suppose that the bit length of key corresponding to the same intermediate value is ξ. For each of 2^{ξ} possible keys in each repetition, if the guessing entropy of guessing key $k^{'}\left(0 \leq k^{'} \leq 2^{\xi} - 1\right)$ is ν, then we define the score of guessing entropy of $k^{'}$ as

$$W_{k^{'}} = 2^{\xi} + 1 - \nu. \tag{1}$$

3.2 Determine the Correct Key from Guessing Entropies

Standaert et al. also defined the first order success rate in [14]. A guessing key is returned in each experiment. If this key is the correct one, then we say that the attack experiment satisfies the first order success. Similar to their definition, we define a successful CPA [3] if the correct key s satisfies

$$s = \underset{k}{argmax} \{W_k\}. \tag{2}$$

Actually, the Eq. (2) is usually satisfied when the guessing entropy of the correct key is close to 1. That's to say, if the number of power traces used in our experiment is n, then the probability

$$\lim_{n\to\infty} Pr[s = \underset{k}{argmax} \{W_k\}] = 1. \tag{3}$$

For each successful repetition, the score of guessing entropy of the correct key is greater than these of wrong guessing keys. So, guessing entropy can be used to distinguish the correct key from the wrong ones. This is why guessing entropy can also contribute to the key recovery.

Let n denote the number of power traces used in each repetition. We define a function of scores of guessing entropies corresponding to guessing key $k^{'}$ on η repetitions $(Exp_1, Exp_2, \cdots, Exp_\gamma)$ as

$$\psi\left(\gamma, n, k^{'}\right) = f\left(W_{k^{'}}^{Exp_1}, W_{k^{'}}^{Exp_2}, \cdots, W_{k^{'}}^{Exp_\gamma}\right). \tag{4}$$

Actually, ψ can be any function of the scores of guessing entropies, such as multiplication, addition, etc. In this paper, we only use a very simple function

$$\psi\left(\gamma, n, k^{'}\right) = \frac{1}{\eta}\sum_{i=1}^{\eta} W_{k^{'}}^{Exp_i}. \tag{5}$$

The limit of function ψ is equal to 2^ξ if the guessing key $k^{'}$ is correct. However, if the guessing key is incorrect, the limit of ψ is equal to $\frac{2^\xi+1}{2}$. That's to say, for each repetition i $(1 \le i \le \eta)$, the following formula 6 is satisfied.

$$\lim_{\gamma\to\infty} \psi(\gamma, n, s) > \lim_{\gamma\to\infty} \psi(\gamma, n, \delta) \quad \delta \in \left\{0, 1, \cdots, 2^\xi - 1\right\} \setminus \{s\}. \tag{6}$$

Formula (6) indicates that when the number of power traces we use in each experiment is large enough, the average score of guessing entropies $\lim_{\gamma\to\infty} \psi(\gamma, n, s)$ of correct key s will be the maximum with a limitation of 2^ξ. The limits of scores of guessing entropies of other wrong guessing keys are $\frac{2^\xi+1}{2}$. That is why the correct key falls in the location of largest score. So as to meet the definition of the first order success rate in [14]. Thus, the feasibility of GE-CPA is well explained in theory.

3.3 Attack Flow

In the above Subsect. 3.2, we show how to recover the key from GE-CPA. Suppose that we randomly encrypt m plaintexts and acquire m power traces. The number of repetitions C is set to 0, the array $W\left[0 \cdots 2^{\xi}-1\right]$ saves the scores of guessing entropies for all possible keys from 0 to $2^{\xi}-1$, and k indicates the number of power traces added in each repetition. The steps to recover the key using GE-CPA are as follows:

Step 1: The attacker randomly selects $n(n << m)$ power traces and the corresponding n plaintexts. He then sets the number of repetitions $C = C + 1$.

Step 2: For each guessing key, he calculates the assumed power consumption using power model(i.e. Hamming distance model). Then, he calculates the correlation coefficients between the assumed power consumption and the power traces. All correlation coefficients are returned, which is different from traditional CPA.

Step 3: The attacker sorts the correlation coefficients, determines the score of guessing entropy for each possible key key_i. Then, he adds them to the array of scores of guessing entropies W. Actually, Other functions like multiplication can also be used.

Step 4: The attacker sorts the array W, determines the score of guessing entropy for each key. If the score of guessing entropy of the correct key is still not the maximum in W, then, $n = n + k$. Otherwise, he goes to step 1.

The greater the score of guessing entropy of the correct key, the higher certainty the key is. The correct key can be recovered with a probability of 1.00 when the score of guessing entropy corresponding to the correct key is equal to 256.

3.4 Success Rate in Guessing Entropy Enhanced CPA

The relationship between different repetitions and guessing entropies has not been taken into consideration in traditional side channel attacks. They just judge whether an experiment is successful. Therefore, in the statistics, the success rate will be close to a fixed value (i.e. 0.50), which not grows with the increase number of repetitions. In our GE-CPA, we consider the relationship between repetitions and guessing entropies. When the number of power traces used in each repetition is sufficient to make the score of guessing entropy corresponding to the correct key larger than those of wrong keys, the success rate grows with the increase number of repetitions. Finally, the success rate of GE-CPA is close to 1. Thus, the success rate of GE-CPA is very different from that of traditional side channel attacks.

Actually, the number of power traces in each repetition is far from infinity. The score of guessing entropy of the correct key is just close to a fixed value larger than $\frac{2^{\xi}+1}{2}$ rather than 2^{ξ} after many repetitions if the number of power traces we use in each repetition is relatively small. That is to say, the score of guessing entropy corresponding to the correct key will be larger than other ones

corresponding to wrong guessing keys. In this case, the correct key can also be distinguished from the wrong ones.

Suppose that we randomly select τ power traces from a total number of N and repeat this operation γ times. The average score of guessing entropy of the correct key is μ $\left(\mu > \frac{2^{\xi}+1}{2}\right)$. The limits of scores of guessing entropies corresponding to the wrong guessing keys are $\frac{2^{\xi}+1}{2}$ when $\gamma \to \infty$. Let $T_{suc}^{\gamma,C_N^{\tau}}$ and $T_{unsuc}^{\gamma,C_N^{\tau}}$ denote the number of successful and unsuccessful repetitions respectively, then

$$\lim_{\gamma\to\infty} T_{suc}^{\gamma,C_N^{\tau}} = \gamma. \tag{7}$$

Then, the success rate ϕ of our GE-CPA is

$$\phi = \lim_{\gamma\to\infty} \frac{T_{suc}^{\gamma,C_N^{\tau}}}{T_{suc}^{\gamma,C_N^{\tau}} + T_{unsuc}^{\gamma,C_N^{\tau}}} = 1 \tag{8}$$

Actually, the variable γ doesn't need to be infinity, the greater the variable γ is, the closer to μ the average score of guessing entropy of the correct key will be. The score of guessing entropy of the correct key is obviously larger than those of wrong keys after many repetitions. γ is less than 30 when only a small number of power traces with success rate ranging from 5 % to 8 % are used. Failure will not happen if more repetitions are done. This also demonstrates the high efficiency of our GE-CPA.

4 Experimental Results

4.1 Experiments on AT89S52 Single Chip

Our first experiment is performed on an AT89S52 single chip. The clock frequency of this chip is 12 MHz. The minimum instructions takes 12 clock cycles for execution. We utilize a *Tektronix DPO* 7254 oscilloscope, and the sampling rate is set to $1GS/s$. The output of S-box in the first round of AES encryption is chosen as the attack point. We test the instruction '$MOVCA, @A + DPTR$', which treats the value of register A as the offset and treats the address $DPTR$ of S-box as the base address, then looks up table S-box and writes the result back to register A (as shown in Fig. 1).

We randomly encrypt 4000 plaintexts and acquire 4000 power traces. Each power trace contains 5000 time samples. We calculate the correlation coefficients between each time sample and Hamming weights of the outputs of S-box. We randomly choose 150 power traces and the corresponding success rate is 93 %. We randomly choose power traces from a total number of 150 and repeat this operation for 200 times to calculate the success rate and guessing entropy (as shown in Fig. 2). The success rate ranges from 1 % to 61 % when $10 \sim 110$ are used. The corresponding guessing entropies range from 115.9 to 1.35 (as shown in Fig. 2(b)). When 40, 60, 80 and 100 power traces are used, the success rates reach 3 %, 7.6 %, 11 % and 16.5 %. The corresponding guessing entropies reach 29.5, 12.5, 5.1 and 2.2 respectively.

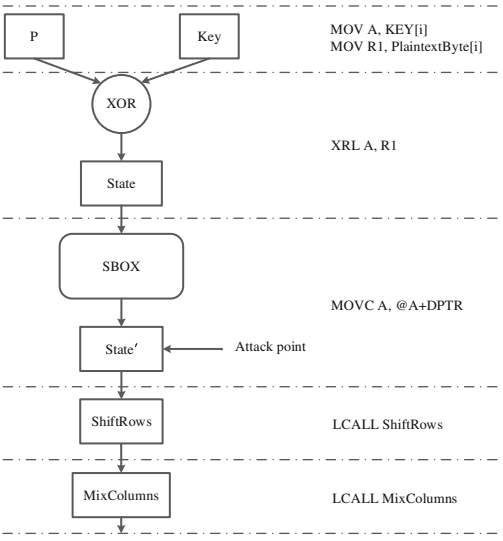


Fig. 1. The output of S-box in the first round of AES algorithm.

The experimental results of our GE-CPA are shown in Fig. 3. The key even can be recovered after only 10 repetitions when 40 power traces with a success rate of 5.5 % are used. This also indicates that our GE-CPA can significantly improve the efficiency of CPA. The average score of guessing entropy of the correct key is close to 226.5, which is stably larger than these corresponding to wrong keys. When more power traces are used in each repetition, the guessing entropy corresponding to correct key decreases, leading to the increase of the corresponding score of guessing entropy.

The guessing entropies are 29.5, 12.5, 5.1 and 2.2 when 40, 60, 80 and 100 power traces are used respectively. With more repetitions, the average score of guessing entropy corresponding to the correct key are close to 226.5, 243.5, 250.9

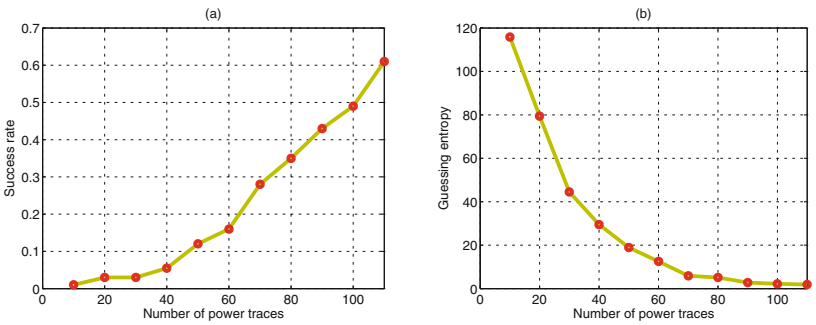


Fig. 2. Success rate (a) and guessing entropy (b) of CPA on AT89S52.

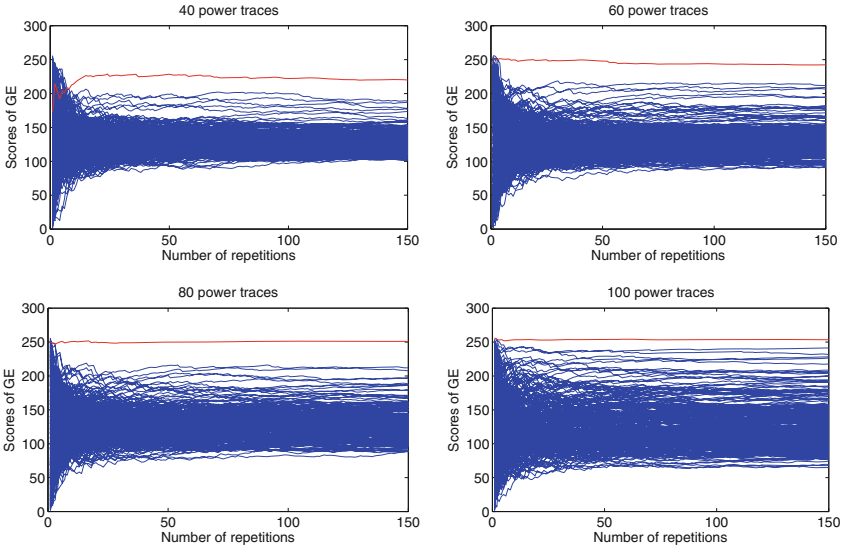


Fig. 3. Score of guessing entropy for each key in different repetitions when 60, 80, 100 and 120 power traces are used in each repetition.

and 253.8 respectively. Other scores of guessing entropies corresponding to wrong keys are relatively smaller.

The experimental results will be better if a small number of power traces are randomly selected from a large power trace set in each repetition. However, this paper tries to solve the problem that how to recover the key with a probability of about 1.00 under the condition that the number of power traces is small, and success rates is low. So, we only consider to use a small number of power traces in our experiment. For example, we use 150 power traces in total in this experiment. If we randomly select 150 power traces from a total number of 4000 to perform CPA, the success rate is equal to 0.93, which is less than 1.00.

4.2 Experiments on SASEBO

Our second experiment is on DES algorithm implemented on the side channel attack standard evaluation board (SASEBO). We use the power trace set of *DPA contest v*1 provided on the website of *DPA contest* [1]. 5000 power traces of power trace set *secmatv*1 are downloaded. We attack the first S-box with 6 bits input and 4 bits output in the last round of the DES algorithm (as shown in Fig. 4). We use the $14000^{th} \sim 16000^{th}$ time samples in our experiments.

We randomly select 200 power traces from a total number of 5000 and repeat the operation 200 times. The first order success rate is about 80 %. Then, we randomly select power traces from a total number of 200 to perform CPA. The success rates range from 1 % to 39 % when $10 \sim 140$ power traces are used (as shown in Fig. 5(a)). The corresponding guessing entropies range from 33.6 to

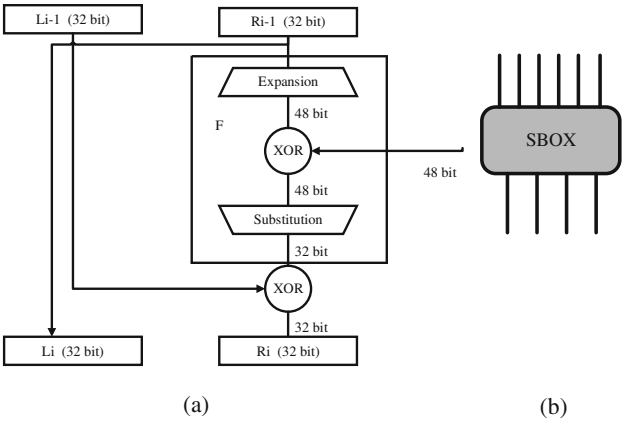


Fig. 4. DES algorithm. (a) shows a round of DES, and (b) shows the S-box of DES.

6.13 (as shown in Fig. 5(b)). When 40, 60, 80 and 100 power traces are used, the success rates reach 3 %, 7.6 %, 11 % and 16.5 % (as shown in Fig. 5(a)). The corresponding guessing entropies are 25.56, 19.4, 16.1 and 10.8 respectively (as shown in Fig. 5(b)).

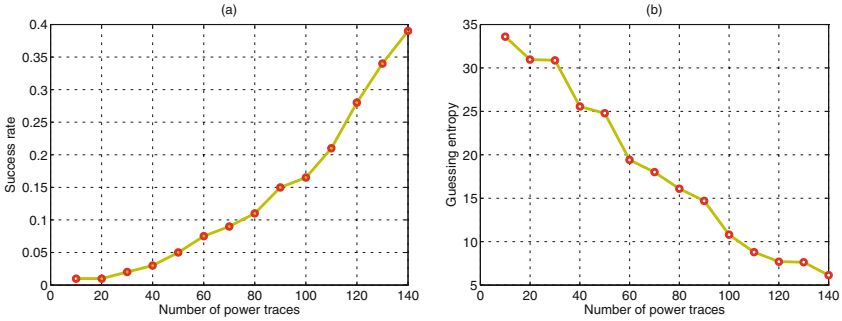


Fig. 5. Success rate (a) and Guessing entropy (b) of CPA on SASEBO.

When more power traces are used, the guessing entropy corresponding to the correct key decreases, leading to the increase of the corresponding score (as shown in Fig. 6). The score of guessing entropy corresponding to the correct key becomes the largest one after 5 repetitions when average 60,100 and 120 power traces are used.

However, in order to obtain a stable success rate, the experiments with average 60 power traces have better results than that with 80 power traces. This also indicates, the guessing entropies of the initial several experiments are very important, which may affect the starting position of success. The score of guessing entropy of the correct key will be greater if more power traces are used

after a few repetitions. However, this doesn't mean the difference of the scores of guessing entropies between the correct key and wrong keys is better. Because of the limited number of repetitions, less power traces used in each repetition may get greater difference.

When 80 power traces are randomly selected, the score of guessing entropy of the correct key becomes stable in different repetitions (as shown in Fig. 7). Sometimes, the experiment has a good beginning (as shown in Fig. 7(a)), the score of guessing entropy of the correct key becomes the maximum after $1 \sim 6$ repetitions. However, it gradually reduce after 7 repetitions. The change is stable after 26 repetitions. The score of guessing entropy of the correct key is sometimes small at the beginning of our experiments. However, it becomes the maximum after many repetitions (as shown in Fig. 7(b)). Regardless of the beginning, the correct key can finally be distinguished from the wrong ones.

It is worth noting that we use a total number of 200 power traces to compare the difference of guessing entropies between the correct key and wrong keys. Actually, when 60 power traces are used in each repetition, our GE-CPA can recover the key (As shown in Fig. 6). Some power traces may be repeatedly selected if the total number of power traces is small and the number of repetitions is large. The effectiveness may be a little worse than randomly selecting power traces from a large power trace set. In addition, the quality of this power trace set may affect the experimental results. However, the experimental results are almost the same if we randomly select 60 power traces from a total number of 120 or 200.

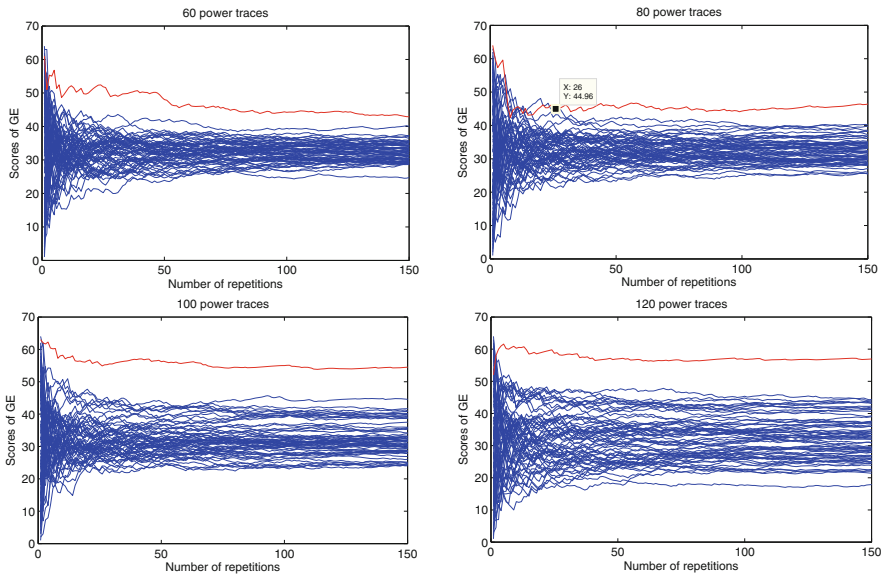


Fig. 6. Scores of guessing entropies for each key in different repetitions when 60, 80, 100 and 120 power traces are used in each repetition.

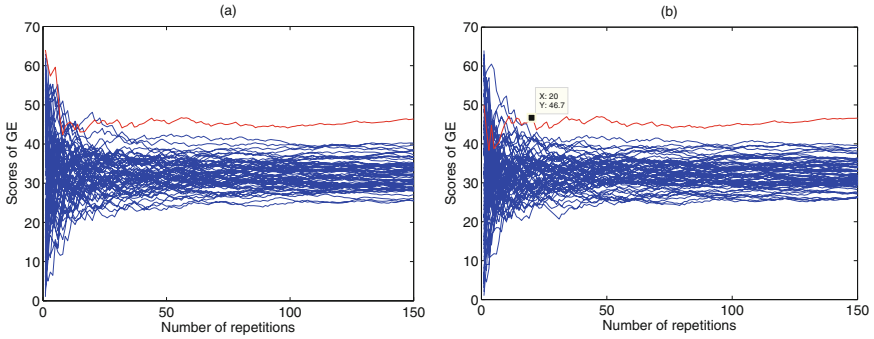


Fig. 7. Scores of guessing entropies for each key in different repetitions when 80 power traces are randomly selected in each repetition.

Komano et al. proposed built-in determined sub-key correlation power analysis, 65 power traces are used to recover the key. Which is more efficiency than our GE-CPA [5]. We also simply compare the efficiency of our guessing entropy enhanced CPA (GE-CPA) with other enhanced side CPA attacks shown on the web site of DPA contest v1 [1]. Hideo used 107 power traces by using his advanced BS-CPA. Yongdae used 119 power traces by using his "CPA with chosen order". Daisuke used 120 power traces by using his "Dual round attack by BS-CPA using improved power model". The efficiency of those attacks are similar to our GE-CPA. Benedikt used 329 power traces by using the difference of means on the last round of DES. Victor used 322 power traces by using his "CPA on the 16th round of the DES selecting the good temporal window". Other attacks using more power traces are not deteiledly introduced here. Our GE-CPA is more efficiency than these attacks.

5 Conclusion

As a common evaluation to evaluate the effectiveness of side channel attacks, guessing entropy is used to measure the uncertainty of the key. In this paper, we analyze the statistical characteristics of guessing entropy and propose GE-CPA. Experiments on AES algorithm implemented on an AT89S52 single chip and power trace set *secmatv*1 of DES algorithm implemented on the side channel attack standard evaluation board (SASEBO) from the website *DPA contest v*1 show that our scheme can efficiently recover key. Our scheme can significantly improve the effectiveness of CPA.

Actually, CPA is just a common type of side channel analysis. Guessing entropy can enhance many other types of side channel analysis like DPA and template attack, etc. This also indicates the practicability of our scheme.

Acknowledgment. This research is supported by the Nation Natural Science Foundation of China (No. 61372062). The authors would like to thank the anonymous referees of IFIP SEC 2016 for the suggestions to improve this paper.

References

1. Dpa contest. http://www.dpacontest.org/home/
2. Backes, M., Köpf, B.: Formally bounding the side-channel leakage in unknown-message attacks. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 517–532. Springer, Heidelberg (2008)
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
4. Fei, Y., Luo, Q., Ding, A.A.: A statistical model for DPA with novel algorithmic confusion analysis. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 233–250. Springer, Heidelberg (2012)
5. Komano, Y., Shimizu, H., Kawamura, S.: Bs-cpa: built-in determined sub-key correlation power analysis. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **93**(9), 1632–1638 (2010)
6. Lomné, V., Prouff, E., Rivain, M., Roche, T., Thillard, A.: How to estimate the success rate of higher-order side-channel attacks. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 35–54. Springer, Heidelberg (2014)
7. Luo, Q., Fei, Y.: Algorithmic collision analysis for evaluating cryptographic systems and side-channel attacks. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 75–80. IEEE (2011)
8. Mather, L., Oswald, E., Whitnall, C.: Multi-target DPA attacks: pushing DPA beyond the limits of a desktop computer. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 243–261. Springer, Heidelberg (2014)
9. Nassar, M., Souissi, Y., Guilley, S., Danger, J.-L.: "Rank Correction": a new side-channel approach for secret key recovery. In: Joye, M., Mukhopadhyay, D., Tunstall, M. (eds.) InfoSecHiComNet 2011. LNCS, vol. 7011, pp. 128–143. Springer, Heidelberg (2011)
10. Nassar, M., Souissi, Y., Guilley, S., Danger, J.-L.: RSM: a small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1173–1178. IEEE (2012)
11. Rivain, M.: On the exact success rate of side channel analysis in the gaussian model. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 165–183. Springer, Heidelberg (2009)
12. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)
13. Souissi, Y., Nassar, M., Guilley, S., Danger, J.-L., Flament, F.: First principal components analysis: a new side channel distinguisher. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 407–419. Springer, Heidelberg (2011)
14. Standaert, F., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques Advances in Cryptology - EUROCRYPT, pp. 443–461, Cologne, Germany, 26–30 April, 2009
15. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition *vs.* comparison side-channel distinguishers: an empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected CMOS devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)

16. Venelli, A.: Efficient entropy estimation for mutual information analysis using B-splines. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 17–30. Springer, Heidelberg (2010)
17. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 390–406. Springer, Heidelberg (2013)

Software Security

Advanced or Not? A Comparative Study of the Use of Anti-debugging and Anti-VM Techniques in Generic and Targeted Malware

Ping Chen(✉), Christophe Huygens, Lieven Desmet, and Wouter Joosen

iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium
{ping.chen,christophe.huygens,
lieven.desmet,wouter.joosen}@cs.kuleuven.be

Abstract. Malware is becoming more and more advanced. As part of the sophistication, malware typically deploys various anti-debugging and anti-VM techniques to prevent detection. While defenders use debuggers and virtualized environment to analyze malware, malware authors developed anti-debugging and anti-VM techniques to evade this defense approach. In this paper, we investigate the use of anti-debugging and anti-VM techniques in modern malware, and compare their presence in 16,246 generic and 1,037 targeted malware samples (APTs). As part of this study we found several counter-intuitive trends. In particular, our study concludes that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware, although targeted malware tend to have a lower antivirus detection rate. Moreover, this paper even identifies a decrease over time of the number of anti-VM techniques used in APTs and the *Winwebsec* malware family.

1 Introduction

In recent years, a new category of cyber threats, known as Advanced Persistent Threat (APT), has drawn increasing attention from the industrial security community. APTs have several distinguishing characteristics which make them quite different from traditional threats [7]. For example, APTs target mostly companies in critical sectors and governmental institutions [11]; the threat actors in APT attacks are highly-organized and well-resourced group, and can even be state-sponsored [17], and they use stealthy techniques, stay low and slow to evade detection.

APT attacks are widely assumed to be more advanced than traditional attacks, mainly because the threat actors are highly organized, working in a coordinated way, and are well-resourced, having a full spectrum of attack techniques. However, it is unclear whether the targeted malware (malware used in APT attacks) are also more advanced than generic malware (malware used in traditional attacks) or not. To better understand APT attacks, we investigate the difference between targeted malware and generic malware, in order to answer the

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 323–336, 2016.
DOI: 10.1007/978-3-319-33630-5_22

research question: *"Is targeted malware more advanced than generic malware?"* In particular, we focus on comparing the usage of anti-debugging and anti-VM techniques in targeted and generic malware.

To defend against malware, defenders have turned to the collection and analysis of malware as mechanisms to understand malware and facilitate detection of malware. In response to this, malware authors developed anti-debugging and anti-VM techniques to avoid being analyzed, hence increasing the difficulty of detection. In this paper, we use the presence of anti-debugging, the presence of anti-VM techniques and the antivirus detection rate as metrics to measure the malware's ability to evade malware analysis and antivirus products. All three measurements can be achieved via static analysis on the malware samples.

By analyzing 1,037 targeted malware samples, as well as 16,246 generic malware samples from 6 different families, we report the usage of anti-debugging and anti-VM techniques in malware. We then compare the presence measurements between targeted and generic malware, and correlate them with their antivirus detection rate, and we examine their evolution over time.

As part of this study we found several counter-intuitive trends. In particular, our study concludes that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware. Moreover, this paper even identifies a decrease over time of the number of anti-VM techniques used in APTs and the *Winwebsec* malware family.

The contributions in this paper are as follows:

- We report on the presence of anti-debugging and anti-VM techniques on 17,283 malware samples, and their associated antivirus detection rate (Sect. 4)
- We analyse and discuss the presence of anti-debugging and anti-VM techniques over time (Sect. 5.2)
- We analyse and discuss the correlation between the presence of anti-debugging and anti-VM techniques and the antivirus detection rate (Sect. 5.3)

2 Overview

2.1 Research Questions

In this paper, we compare the targeted malware and generic malware by investigating the following research questions:

Q1: Does targeted malware use more anti-debugging techniques?
Q2: Does targeted malware use more anti-VM techniques?
Q3: Does targeted malware have lower antivirus detection rate?

Since APT attacks are more advanced and sophisticated, one might expect that the targeted malware (the weapons of APT attacks) may use more anti-debugging and anti-VM techniques to evade defensive analysis, and have lower antivirus detection rate. We describe the details about these three metrics in Sect. 3, and present the analysis result on these questions in Sect. 4.

Additionally, we are interested about the evolution of the usage of anti-debugging and anti-VM techniques, and how does the use of anti-debugging and anti-VM techniques impact antivirus detection. More specifically, we test the following hypotheses:

H1a: The use of anti-debugging techniques in malware is increasing over time
H1b: The use of anti-VM techniques in malware is increasing over time
H2a: The use of anti-debugging techniques has negative effect on antivirus detection
H2b: The use of anti-VM techniques has negative effect on antivirus detection

While defenders put more and more effort to fight against malware, we assume malware authors are using more and more anti-debugging and anti-VM techniques to thwart the defense, in other words, the use of anti-debugging and anti-VM techniques in malware might increase over years. And the use of these evasive techniques might help malware to evade some antivirus products. To test the hypotheses, we present correlation analysis in Sect. 5.

2.2 Dataset

The targeted malware samples used in our study are collected from various publicly available reports on APT attacks [9,14,15,17,19,24]. These reports are published by security companies such as FireEye and Kaspersky, to provide technical analysis over various APT attacks, and they typically include the hashes of the discovered targeted malware. With the malware hashes, we then use VirusTotal Private API [2] to search and download these samples.

In this way, we collected 1,037 targeted malware samples ranging from 2009 to 2014. The date information of a malware is extracted from the timestamp attribute in a PE file. For our comparative study, a dataset of more than 16,000 malware samples that belong to 6 generic malware families was collected from VirusTotal. For each malware family and each year (from 2009 to 2014), we use VirusTotal Private API to search for maximum 600 malware samples.

Compared to targeted malware, these malware families are more popular and well understood in the industry. The number of samples belonging to each malware family and the corresponding brief description are shown in Table 1.

3 Metrics

In this paper, we use the presence of anti-debugging, the presence of anti-VM techniques and the antivirus detection rate as metrics to measure the malware's ability to evade malware analysis and antivirus products.

We only focus on these three metrics that can be detected through static analysis. While there are other metrics that can be used to measure the sophistication of malware, such as stealthy network communication, self-deleting execution, they require executing the malware in a targeted environment. Since it is difficult to determine the targeted environment for executing malware, we leave out the dynamic analysis.

Table 1. Overview of malware dataset

Malware family	Discovered year	# of samples	Brief description
Sality	2003	2926	general and multi-purpose [5]
Zbot	2007	3131	banking trojan
Winwebsec	2009	2741	rogueware, fake antivirus [6]
Ramnit	2010	2950	information stealer [4]
Zeroaccess	2011	1787	botnet, bitcoin mining [22]
Reveton	2012	1711	ransomware [25]
Targeted (APT)	2009	1037	targeted malware

3.1 Anti-debugging Techniques

In order to thwart debuggers, malware authors use anti-debugging techniques to detect the presence of debuggers and compromise the debugging process. There are many anti-debugging techniques, we focus on detecting the anti-debugging techniques that are known in literature [10]. Since the complete list of anti-debugging techniques is too verbose, we only show those are detected in our malware dataset, as shown in Table 2.

The anti-debugging techniques that are found in our study can be categorized into three types. The use of Windows APIs is the easiest way to detect a debugger. Additionally, malware can check several flags within the PEB (Process Environment Block) structure and the process' default heap structure to detect debuggers. The third way to use some instructions that trigger characteristic behavior of debuggers (e.g., use `RDTSC` to measure execution time).

Table 2. Popular anti-debugging techniques

Type	Name
Windows APIs	`IsDebuggerPresent, SetUnhandledExceptionFilter, FindWindow, CheckRemoteDebuggerPresent, NtSetInformationThread, NtQueryInformationProcess, GetProcessHeap, GetTickCount, NtQuerySystemInformation, OutputDebugString, BlockInput, QueryPerformanceCounter, VirtualProtect, SuspendThread, WaitForDebugEvent, SwitchDesktop, CreateToolhelp32Snapshot`
Flags	`PEB fields (NtGlobalFlag, BeingDebugged)` `Heap fields (ForceFlags, Flags)`
Instructions	`RDTSC, RDPMC, RDMSR, ICEBP, INT3, INT1`

To detect these anti-debugging techniques in a malware sample, we first look for the Windows APIs in the import address table (IAT) of the PE file. Next, we use IDA [1] to automatically disassemble the sample and generate an assembly listing file, and then search for the specific instructions in the assembly listing file to detect the use of flags and instructions. If any of these techniques are found in the IAT or the assembly listing file, we consider the malware sample use anti-debugging techniques.

3.2 Anti-VM Techniques

There are mainly three types of VM detection techniques [20,21]: (1) Interaction based. Sandboxes emulate physical systems, but without a human user. Malware detects VM by checking common human interactions such as mouse movement and mouse clicks. (2) Artifacts based. Virtual machines may have unique artifacts such as service list, registry keys, etc. And some CPU instructions such as `SIDT` have characteristic results when executed inside virtual machines. Malware can leverage these differences to detect sandboxing environment. (3) Timing based. Due to the large number of file samples to examine, sandboxes typically monitor files for a few minutes. Malware authors can configure the malware to execute only after some sleeps, or after a given date and time, in order to avoid being analyzed. Table 3 shows the anti-VM techniques that are found in our malware samples. Details about these techniques can be found in [20,21].

Table 3. Popular anti-VM techniques

Type	Name
Windows APIs	`GetCursorPos, Sleep, NtOpenDirectoryObject, NtEnumerateKey` `GetSystemFirmwareTable, NtQueryVirtualMemory, NtQueryObject`
Instructions	`SIDT, SLDT, SGDT, STR, IN, SMSW, CPUID`
Strings	`'sbiedll.dll', 'dbghelp.dll', 'vmware'`

To detect these anti-VM techniques in a malware sample, we follow the same method for detecting anti-debugging techniques. Additionally, we extract strings from a PE file, in order to search for the specific strings. If any of these techniques are found, we consider the malware sample use anti-VM techniques.

3.3 Antivirus Detection Rate

Since the adoption of AV products, malware authors are consistently trying to evade them. There are various techniques and tools [18] to build malware that can bypass common AV products.

In this paper, we use antivirus detection rate as a metric to compare malware's ability to bypass AV products. We get the detection results from 56 AV engines provided in VirusTotal. Since AV engines frequently update their signatures in order to detect malware samples that are not previously spotted by their engines, the reports in VirusTotal might not reflect the current status of these AV engines. To compare the AV detection rate of different malware families, we rescanned all malware samples within two days in December 2014 by using VirusTotal's API [2] to get the most recent detection results.

4 General Findings

4.1 The Usage of Anti-debugging and Anti-VM Techniques

To answer questions **Q1, Q2**, we first calculate the percentage of samples that use anti-debugging and anti-VM techniques for each malware family. As shown in Table 4, the majority of samples use either anti-debugging or anti-VM techniques. 68.6 % targeted malware samples use anti-debugging techniques, which is less than most generic malware families, and 84.2 % targeted malware samples use anti-VM techniques, which is more than all generic malware families. Thus by simply comparing the percentage of samples in each family, we can see that anti-debugging techniques are less popular in targeted malware, and anti-VM techniques are more commonly found in targeted malware.

Table 4. Percentage of samples using anti-debugging/anti-VM techniques in each malware family

Family	% Anti-debug.	% Anti-VM	Family	% Anti-debug.	% Anti-VM
Sality	89.6 %	76.2 %	Ramnit	85.8 %	71.6 %
Zbot	72.9 %	39.7 %	Zeroaccess	41.6 %	50.4 %
Winwebsec	80.0 %	52.9 %	Reveton	74.8 %	62.8 %
Targeted (APT)	68.6 %	84.2 %			

We then calculate the average the number of detected anti-debugging/anti-VM techniques in each family, and compare the average numbers of generic malware family to targeted malware using Z-test. Z-test is a statistical function that can be used to compare means of two samples. With a Z value bigger than 2.58 and p-value smaller than 1 %, we can say that the means of two samples are significantly different.

As shown in Table 5, the average number of detected anti-debugging techniques in targeted malware is neither smallest nor biggest. Since all the Z values are bigger than 2.58, with p-values smaller than 1 %, we can accept all the hypotheses that the average number of detected anti-debugging in targeted malware is significantly different to all generic malware families. In other words, targeted malware do not necessarily use more anti-debugging techniques than generic malware. Thus the answer to question **Q1** is still negative.

As for the use of anti-VM techniques, it is the same case as the use of anti-debugging techniques. Targeted malware do not necessarily use more anti-VM techniques than generic malware. Hence the answer to question **Q2** is also negative. The results can be better illustrated in box plots. As shown in Fig. 1, the average number of anti-debugging/anti-VM techniques in targeted malware is less than some generic malware family.

Table 5. Average number of anti-debugging/anti-VM techniques in each family

Malware Family	Anti-debugging		Anti-VM	
	# of techniques	Z value, p-value	# of techniques	Z value, p-value
Sality	3.59	$11.4, 4.17 \times 10^{-30}$	1.25	$8.4, 3.60^{-17}$
Zbot	2.05	$6.2, 6.34 \times 10^{-10}$	0.48	$15.9, 5.83^{-57}$
Winwebsec	1.75	$11.4, 2.63 \times 10^{-30}$	0.71	$8.8, 1.06^{-18}$
Ramnit	3.76	$13.3, 2.57 \times 10^{-40}$	1.30	$10.2, 1.57^{-24}$
Zeroaccess	0.96	$20.3, 2.27 \times 10^{-91}$	0.17	$40.0, 0$
Reveton	1.78	$7.5, 4.89 \times 10^{-14}$	0.48	$15.2, 2.45^{-52}$
Targeted (APT)	2.57	Not Applicable	0.94	Not Applicable

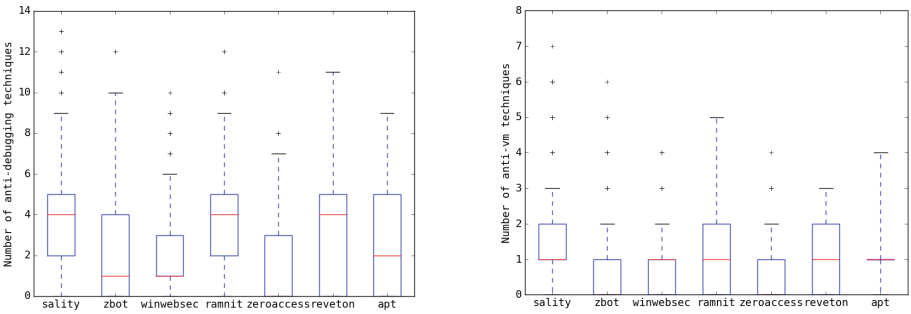


Fig. 1. Number of detected anti-debugging/anti-VM techniques in each sample in each family

4.2 Antivirus Detection Rate

To answer question **Q3**, we calculate the average number of antivirus detections from 56 AV scanners for each malware family, and then compare the average numbers of generic malware family to targeted malware using Z-test. As shown in Table 6, targeted malware has the smallest average number of antivirus detections. And the Z-test results shows that all the Z values are bigger than 2.58, with p-value smaller than 1 %. So we accept the hypothesis that targeted malware has significant lower antivirus detections than generic malware, and the the answer to question **Q3** is positive.

Table 6. Average number of antivirus detections for each malware family

Family	# detections	Z value, p-value	Family	# detections	Z value, p-value
Sality	45.7	20.2, 1.35×10^{-90}	Ramnit	49.8	37.8, 0
Zbot	44.6	15.3, 8.49×10^{-50}	Zeroaccess	47.9	36.0, 3.04^{-284}
Winwebsec	46.7	35.5, 4.39×10^{-276}	Reveton	44.5	16.5, 1.71^{-61}
APT	39.5	Not Applicable			

To better illustrate the result, we made a box plot to show the number of AV engines that detected each sample in each family (in Fig. 2). We can clearly observe that targeted malware have a lower antivirus detection rate, compared to generic malware. Figure 2 shows that all box plots have long whiskers (with the exception of the Reveton malware), which indicates some malware samples (0.8 %) are only detected by a few antivirus engines (less than 10).

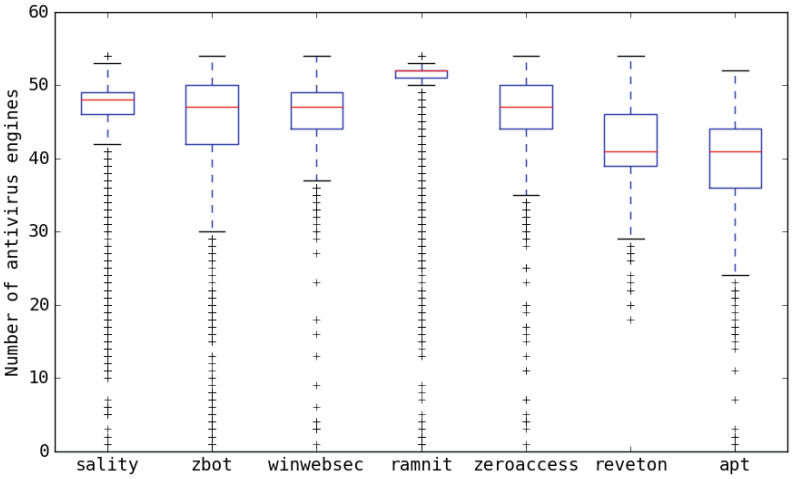


Fig. 2. Number of AV engines that detected each sample in each family

As for the evolution of antivirus detection rate (in Fig. 3), we can observe that the detection rate tends to decrease over years. This is because malware samples that have been discovered earlier are well populated in antivirus repositories, and being analyzed more often than newly discovered malware samples. Compared to generic malware, targeted malware samples have lower detections for most of the time. We would expect older malware samples to have high detections, but there are still about 13 antivirus engines that cannot detect targeted malware that already discovered in 2009 and 2010.

5 Correlation Analysis

In order to test hypothesis **H1a, H1b, H2a, H2b**, we use Spearman's rank correlation to investigate the evolution of the use of anti-debugging and anti-VM techniques, and the correlation between the use of anti-debugging (or anti-VM) techniques and antivirus detection rate.

5.1 Spearman Correlation

Spearman's rank correlation coefficient is a nonparametric measure of the monotonicity of the relationship between two variables. It is defined as the

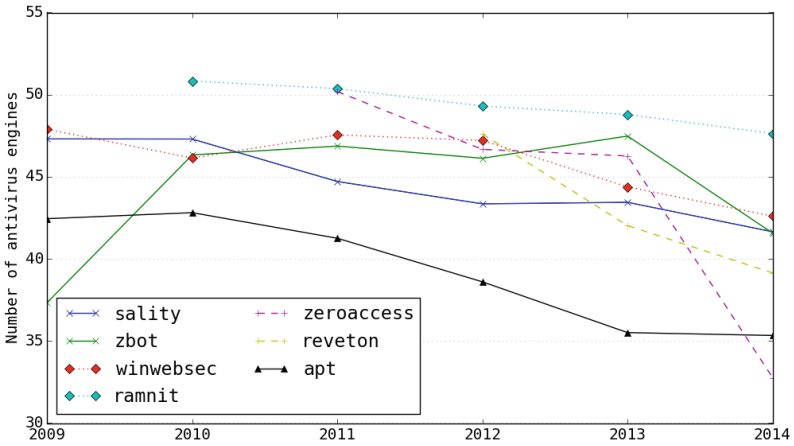


Fig. 3. Evolution of antivirus detection rate

Pearson correlation coefficient between the ranked variables. However, unlike the Pearson correlation, the Spearman correlation does not assume that both variables are normally distributed. It is a nonparametric statistic, which do not rely on assumptions that the dataset is drawn from a given probability distribution. The result of Spearman correlation varies between -1 and $+1$, and a positive coefficient implies that as one variable increases, the other variable also increases and vice versa. When using Spearman correlation to test statistical dependence, we set the significance level to 5 %. The p-value is calculated using Student's t-distribution. We accept the hypothesis only if the p-value is smaller than the significance level.

5.2 Evolution of the Use of Anti-debugging and Anti-VM Techniques

To test hypothesis **H1a, H1b**, we use Spearman correlation to measure the correlation between the number of anti-debugging/anti-VM techniques found in malware and the time when the malware is created. The build date of a malware sample is extracted from the timestamp attribute in the PE file. While the timestamp attribute might be incorrect, since malware authors can set arbitrary value for it, there is little incentive for them to do this.

Table 7 shows the Spearman correlation coefficient and p-value for each malware family. We can observe that there is positive correlation for most malware families, which implies that malware authors tend to use more and more anti-debugging techniques over years. The *Winwebsec* and *Zbot* family also have a positive correlation coefficient, but the p-values are bigger than the significance level, thus we reject the hypothesis for *Winwebsec* and *Zbot* family.

While for the use of anti-VM techniques, only four malware families have a positive correlation coefficient, the others do not show a positive correlation between the use of anti-VM techniques and build date. The *Winwebsec* and APT family have negative correlation coefficients, and the p-values are smaller than

the significance level, which implies that the use of anti-VM techniques decreases over years. We think this decrease may be attributed to the great increase in the usage of virtualization. Some malware authors are starting to realize that the presence of a virtual machine does not necessarily mean the malware is being analyzed, since more and more organizations are adopting virtualization technology.

Table 7. Spearman correlation between the use of anti-debugging (or anti-VM) techniques and build date

Malware Family	anti-debugging vs. time		anti-VM vs. time	
	coefficient	p-value	coefficient	p-value
Sality	0.23	9.1×10^{-38}	0.31	1.9×10^{-66}
Zbot	0.31	0.08	−0.01	0.39
Winwebsec	0.02	0.36	−0.43	6.7×10^{-129}
Ramnit	0.29	6.1×10^{-62}	0.26	1.7×10^{-48}
Zeroaccess	0.56	4.3×10^{-149}	0.52	8.2×10^{-127}
Reveton	0.45	2.2×10^{-85}	0.54	2.1×10^{-129}
Targeted (APT)	0.29	1.1×10^{-20}	−0.26	4.6×10^{-16}

To better illustrate the evolution of the use of anti-debugging and anti-VM techniques, we group malware samples by the year in which they are compiled and then calculate the percentage of samples using anti-debugging (or anti-VM) techniques in each group. As shown in Figs. 4 and 5, the percentage of samples using anti-debugging techniques in APT malware tend to go up, while the percentage of samples using anti-VM techniques decrease over years. The evolution trends are consistent with the Spearman correlation coefficients in Table 7.

5.3 Correlation Between the Use of Anti-debugging (or Anti-VM) Techniques and Antivirus Detection Rate

To test hypothesis **H2a, H2b**, we use Spearman correlation to measure the correlation between the number of anti-debugging (or anti-VM) techniques found in malware and the number of positive detections. As shown in Table 8, most malware families (except the *Winwebsec* malware) show negative correlation between the use of anti-debugging techniques and antivirus detection rate, which implies that the use of anti-debugging techniques might help malware to evade antivirus products. While for the use of anti-VM techniques, there are four malware families having a negative correlation coefficient. The *Winwebsec* and APT malware show positive correlation between the use of anti-VM techniques and antivirus detection rate, this might due to the decreasing use of anti-VM techniques in both families, as shown in the previous section.

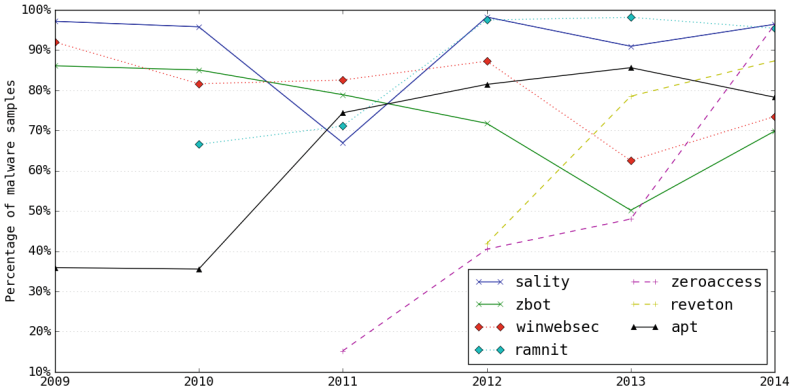


Fig. 4. Evolution of the use of anti-debugging techniques

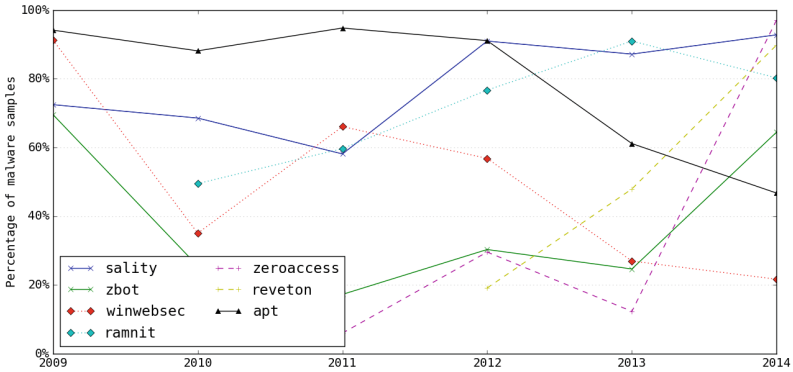


Fig. 5. Evolution of the use of anti-VM techniques

Table 8. Spearman correlation between the use of anti-debugging (or anti-VM) techniques and antivirus detection rate

Malware Family	detection rate vs. anti-debugging		detection rate vs. anti-VM	
	coefficient	p-value	coefficient	p-value
Sality	-0.1	8.1×10^{-9}	-0.07	5.2×10^{-5}
Zbot	-0.17	3.3×10^{-22}	-0.20	3.8×10^{-30}
Winwebsec	0.05	0.004	0.29	3.7×10^{-57}
Ramnit	-0.13	1.6×10^{-13}	0.004	0.80
Zeroaccess	-0.63	1.1×10^{-198}	-0.61	8.7×10^{-183}
Reveton	-0.22	7.2×10^{-21}	-0.30	3.5×10^{-37}
Targeted (APT)	-0.26	1.2×10^{-16}	0.13	1.6×10^{-5}

5.4 Summary

We summarize the hypotheses testing results in Table 9. For the use of anti-debugging techniques, hypothesis **H1a** and **H2a** are accepted for targeted malware and most generic malware (except the *Winwebsec* and *Zbot* family), which indicates that both targeted and generic malware are increasing use anti-debugging techniques and the use of anti-debugging techniques might help malware to evade antivirus products.

For the use of anti-VM techniques, we observe two different trends. Some malware families (*Sality, Ramnit, Reveton*) accept hypothesis **H1b** and **H2b**, while targeted malware and *Winwebsec* malware reject hypothesis **H1b** and **H2b**. There are two possible explanation for the decreasing usage of anti-VM techniques in targeted malware and *Winwebsec* malware: (1) Some targeted machines are increasingly using virtualization technology, thus malware authors discard anti-VM techniques in order to target these machines. (2) Malware authors are using new anti-VM techniques which we cannot detect.

Table 9. Hypotheses testing results with Spearman correlation

Family	**H1a**	**H1b**	**H2a**	**H2b**	Family	**H1a**	**H1b**	**H2a**	**H2b**
Sality	A	A	A	A	Ramnit	A	R	A	NA
Zbot	NA	NA	A	A	Zeroaccess	A	A	A	A
Winwebsec	NA	R	R	R	Reveton	A	A	A	A
APT	A	R	A	R					

A: Accepted, NA: Not Accepted due to a bigger p-value
R: Rejected, the opposite hypothesis is accepted

6 Related Work

APT Attacks. Research on targeted attacks and APTs are mostly from industrial security community. Security service providers (e.g., FireEye, Symantec) periodically publish technical reports that various APT attacks [9,14,15,17,19,24]. Recently, this topic also become hot in academia. In [23], Thonnard et al. conducted an in-depth analysis of 18,580 targeted email attacks, showing that a targeted attack is typically a long-running campaign highly focusing on a limited number of organizations. In [16], Le Blond et al. presented an empirical analysis of targeted attacks against a Non-Governmental Organization (NGO), showing that social engineering is an important component of targeted attacks.

Giura and Wang [12] introduced an attack pyramid model to model targeted attacks, and implemented a large-scale distributed computing framework to detect APT attacks. Hutchins et al. [13] proposed a kill chain model to track targeted attack campaigns and proposed an intelligence-driven strategy to adapt defense based on the gathered intelligence.

Anti-debugging and Anti-VM in Malware. Chen et. al. developed a detailed taxonomy for anti-debugging and anti-VM techniques [8], and they also

proposed a novel defensive approach to mislead the attacker, by disguising the production systems as monitoring systems. A recent survey of the use of anti-debugging and anti-VM techniques in malware is presented by Branco et. al. [3], in which they introduced various static detection methods for anti-debugging and anti-VM techniques, and run an analysis over 4 million samples to show the state of evasion techniques in use.

7 Conclusion

In this paper, we have analyzed the presence of anti-debugging and anti-VM techniques in 17,283 malware samples, by using static analysis. As part of this analysis, we have compared the presence measurements between targeted and generic malware, we have correlated them with their antivirus detection rate, and we have examined their evolution over time.

As expected, we have observed that both targeted malware and generic malware often use anti-debugging and anti-VM techniques. The analysis results also confirmed the hypotheses that the number of anti-debugging techniques used tend to increase over years, and that their presence has a negative correlation with the antivirus detection rate.

At the same time, this study revealed two counter-intuitive trends: (1) The study concluded that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware, whereas targeted malware tend to have a lower antivirus detection rate; (2) This paper identified a decrease over time of the number of anti-VM techniques used in APTs and the winwebsec malware family. This conflicts with the original hypothesis that APTs try to evade analysis and detection by using anti-VM techniques, and strongly contrasts with other malware families where the opposite trend holds.

Acknowledgements. We would like to thank VirusTotal for providing us a private API, and the anonymous reviewers for their comments. This research is partially funded by the Research Fund KU Leuven, iMinds, IWT, and by the EU FP7 projects WebSand, NESSoS and STREWS. With the financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE).

References

1. IDA. https://www.hex-rays.com/products/ida/
2. VirusTotal Private API. https://www.virustotal.com
3. Branco, R.R., Barbosa, G.N., Neto, P.D.: Scientific but not academical overview of malware Anti-debugging, Anti-disassembly and Anti-VM. In: Blackhat (2012)
4. Microsoft Malware Protection Center. Win32/Ramnit. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Ramnit
5. Microsoft Malware Protection Center. Win32/Sality. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Sality
6. Microsoft Malware Protection Center. Win32/Winwebsec. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Winwebsec

7. Chen, P., Desmet, L., Huygens, C.: A study on advanced persistent threats. In: De Decker, B., Zúquete, A. (eds.) CMS 2014. LNCS, vol. 8735, pp. 63–72. Springer, Heidelberg (2014)
8. Chen, X., et al. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In: IEEE International Conference on Dependable Systems and Networks, pp. 177–186 (2008)
9. Cylance. Operation Cleaver (2014)
10. Peter Ferrie. The Ultimate Anti-Debugging Reference (2011)
11. FireEye: FireEye Advanced Threat Report: 2013 (2014)
12. Giura, P., Wang, W.: Using large scale distributed computing to unveil advanced persistent threats. Science 1(3) (2013)
13. Hutchins, E.M., et al.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In: Proceedings of the 6th International Conference on Information Warfare and Security (2013)
14. Kaspersky: The Icefog APT: A Tale of Cloak and Three Daggers (2013)
15. Kaspersky: Energetic Bear - Crouching Yeti (2014)
16. Le Blond, S., Uritesc, A., Gilbert, C., Chua, Z.L., Saxena, P., Kirda, E.: A look at targeted attacks through the lense of an NGO. In: Proceedings of the 23rd USENIX Conference on Security Symposium, pp. 543–558. USENIX Association (2014)
17. Mandiant: APT1: Exposing One of China's Cyber Espionage Unit (2013)
18. Mohanty, D.: Anti-Virus Evasion Techniques Virus Evasion Techniques Virus Evasion Techniques and Countermeasures. http://repo.hackerzvoice.net/depot_madchat/vxdevl/papers/vxers/AV_Evasion.pdf
19. Arbor Networks: Illuminating the Etumbot APT Backdoor (2014)
20. Rin, N.: Virtual Machines Detection Enhanced (2013). http://artemonsecurity.com/vmde.pdf
21. Singh, A., Zheng, B.: Hot Knives Through Butter: Evading File-based Sandboxes (2014)
22. Symantec: Trojan.Zeroaccess. http://www.symantec.com/security_response/writeup.jsp?docid=2011-071314-0410-99
23. Thonnard, O., Bilge, L., O'Gorman, G., Kiernan, S., Lee, M.: Industrial espionage and targeted attacks: understanding the characteristics of an escalating threat. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 64–85. Springer, Heidelberg (2012)
24. Villeneuve, N., et al.: Operation Ke3chang: Targeted Attacks Against Ministries of Foreign Affairs (2013)
25. Wikipedia: Ransomware -Reveton. http://en.wikipedia.org/wiki/Ransomware#Reveton

NativeProtector: Protecting Android Applications by Isolating and Intercepting Third-Party Native Libraries

Yu-Yang Hong(✉), Yu-Ping Wang, and Jie Yin

National Laboratory for Information Science and Technology,
Tsinghua University, Beijing 10084, China
hyy13@mails.tsinghua.edu.cn, {wyp,yinjie}@mail.tsinghua.edu.cn

Abstract. An increasing number of Android developers are incorporating third-party native libraries in their applications for code reuse, CPU-intensive tasks and other purposes. However current Android security mechanism can not regulate the native code in applications well. Many approaches have been proposed to enforce security of Android applications, but few of them involve security of the native libraries in Android applications.

In this paper, we propose NativeProtector, a system that regulates the third-party native libraries in Android applications. The standalone Android application is separated into two components: the server app and the client app where server app contains the native libraries for providing services from the native libraries while the client app contains the rest parts of the original app. The client app binds to the server app at the launching time, and all native function calls are replaced with interprocess calls to the server app. NativeProtector also generates the stub libraries intercept system calls in server app and enforce security of the native libraries in server app. We have implemented a prototype of NativeProtector. Our evaluation shows that NativeProtector can successfully detect and block the attempts of performing dangerous operations by the third-party native libraries in Android applications. The performance overhead introduced by NativeProtector is acceptable.

Keywords: Android security · Native libraries · Process isolation · Call interception

1 Introduction

Android dominated the smartphone market with a share of 82.8 % in the second quarter of 2015 [6]. This trend is benefited from the great increase of third-party Android applications, because they can be easily downloaded and installed. However, numbers of malicious applications also occur to leak user private information and perform dangerous operations. Therefore, preventing privacy leaks and enabling fine-grained control in Android applications are necessary.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 337–351, 2016.
DOI: 10.1007/978-3-319-33630-5_23

Many approaches have been proposed to protect the security of Android applications, but they often focus on Java code, because Android applications are often written in Java language. In fact, Android also provides JNI (Java Native Interface) for calling native libraries in applications, and many developers tend to use third-party native libraries to reuse existing code or perform CPU-intensive tasks (such as image filtering and video encoding). However, the security of these third-party native libraries is often omitted [19]. In the Android system, these native libraries can access the entire process address space and share all the permissions which the user grants to the whole applications, and they are also uncovered by Java security mechanism. Thus, malicious native libraries are very dangerous for Android security.

To the best of our knowledge, only a few existing approaches focus on the security of native libraries in Android. NativeGuard [21] is a typical framework which uses process isolation to sandbox native libraries of applications. It has two main advantages. Firstly, NativeGuard separates native libraries to another standalone application, so native libraries can not fully access the entire application address space, and the interaction between native libraries and Java code is fulfilled via Android Inter-Process Communication (IPC) mechanism. Secondly, the generated native-library application is no longer granted permissions, so dangerous operations can not be performed.

However, NativeGuard still has some limitations. Firstly, because no permissions are granted to the native-library application, the benign native libraries crash when they need necessary permissions. Secondly, NativeGuard lacks fine-grained control of the native libraries to manager their behaviors.

To ensure the security of native libraries in Android, we propose a practical approach named NativeProtector. On one hand, inspired by NativeGuard, we use the process isolation to prevent the native libraries from accessing the entire application address space and limit the permissions of native libraries. On the other hand, we perform fine-grained control of native libraries by instrumenting the third-party native libraries and intercepting native-library calls to access private data and perform dangerous system calls. In detail, the third-party native libraries are separated as a standalone application, so the access of native libraries to Java code is restricted by fine-grained access control. *By combining isolation and interception, we can ensure the security of native libraries without crashing benign native libraries.* Meanwhile, NativeProtector is very easy to deploy. It can run as a common application without the root privilege because NativeProtector statically instruments the target application. Hence, no modification is required for the Android framework.

The main contributions of this paper are following:

- By combining isolation and interception, we have proposed a practical approach named NativeProtector, to protect Android applications from malicious third-party native libraries.
- We have built a prototype of NativeProtector to separate an application to the native-library application and Java code, and instrument the native libraries to perform fine-grained access control.

– We have evaluated NativeProtector on real-world and manually crafted applications. The experimental results show that NativeProtector is effective for security and compatible for many applications, and the performance overhead is also acceptable.

The rest of this paper is organized as follows: Sect. 2 provides some background information on Android security and JNI, we also talk about dynamic loading and linking in Android. In Sect. 3 we describe the threat model of NativeProtector and defenses provided by NativeProtector. Section 4 goes through details about NativeProtector's implementation. In Sect. 5 we evaluate effectiveness, compatibility and overhead of NativeProtector. Related work is shown in Sect. 6. Section 7 gives the conclusion of this paper.

2 Background

In this section, we first briefly give an overview of Android security, and then introduce some important concepts in Android to help to better understand NativeProtector.

2.1 Android Security Overview

Android OS is an open-source software stack for mobile devices consisting of a Linux kernel, Android application framework and system applications. In Android, each application runs in a separate sandboxed environment that isolates data and code from other applications which is guaranteed by Linux kernel's process isolation. One application can not access to another application's address space and private data. Inspired by this mechanism, the third-party native libraries can be separated as another application to ensure that they can not access to the entire application's address space or private data.

2.2 Java Native Interface

Similar to the desktop Java program, Android provides the Java Native Interface (JNI) to define a framework for Java code and native code to call each other. Commonly, developers use native libraries in their applications for code reuse, CPU-intensive task or application hardening. Android provides Native Develop Kit (NDK) [2] to allow developers to implement parts of applications in native languages like C and C++. NDK compiles native source code files into shared libraries which can be loaded dynamically under the request of the application's java code. When a native function is invoked, it will be passed a special data structure of type JNIEnv, which allows the native code to interact with the java code [20]. For instance, the native code can use JNIEnv->FindClass("Sample") to locate the Java class "Sample" and call its functions. Inspired by this mechanism, these key JNI related functions can be interposed to intercept the access to private data and dangerous function calls based on predefined policies.

2.3 Dynamic Loading and Linking

Android adopts its own dynamic loader and linker for native libraries. Unlike the desktop Linux operating system, Android's loader do not take the lazy Binding policy, which means the loader will recursively resolve all the external functions when the application is loading into the memory. But the PLT/GOT [7] structure is still used for dynamic linking. In particular, for an ELF file which has some external functions, its call sites to an external function is actually jump instruction to a stub function in the Procedure Linkage Table (PLT). This stub function performs a memory load on a entry in the Global Offset Table (GOT) to retrieve the real target address of this function call. When a native library is loaded, the loader resolves all external functions and fills them in the GOT entries.

3 System Design

In this section, we first give the threat model of NativeProtector, and then we explain how can we prevent the damage from the third-party native libraries and the defenses provided by NativeProtector.

3.1 Threat Model

There are three main advantages for using native libraries or native code in Android applications: a) porting applications between platforms; b) reusing existing libraries or providing libraries for reuse; c) increasing performance in certain cases, particularly for CPU-intensive applications like pixel rendering, image filtering and games. For these advantages, many Android developers tend to incorporate native libraries in their applications. But these native libraries are always developed by the third party, so they can be malicious. Note that a carefully designed Android application may have security check on passing sensitive data to native library. However, the developers usually trust the native libraries and fail to perform such checks.

Similar to existing solutions [21], we assume an adversary model that the third-party native libraries in the applications are not trustworthy, but some native libraries are essential to the functionalities of the applications. We need to ensure the applications with third-party native libraries work as normally as possible when restricting the third-party native libraries' unprivileged operations such as accessing to the private data.

In this paper, the developer is trustworthy so that the Java code of the applications is trustworthy. As there are many approaches of enforcing the security of Java code in Android applications [13,16,25,26], it's reasonable that the Java code can be regulated well even though it involving some malicious code.

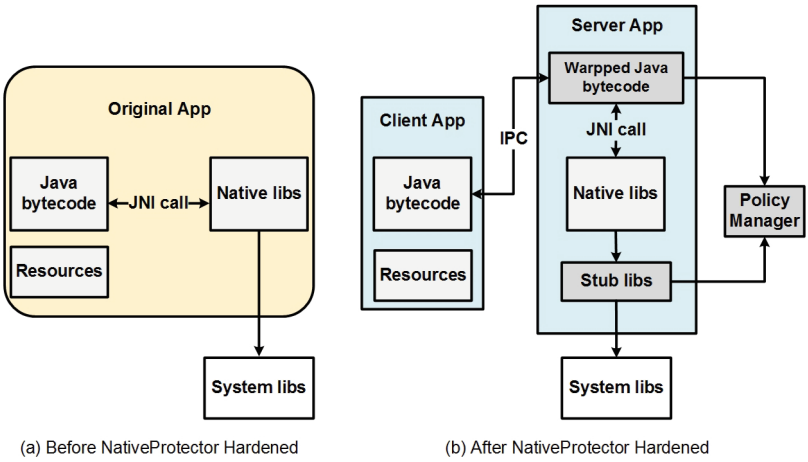


Fig. 1. System architecture of NativeProtector

3.2 Defenses Provided

As shown in Fig. 1, NativeProtector separates the original app to two standalone apps. One of them consists of Java bytecode and the resources of the original app (like layouts, values et al.). The other consists of the third-party native libraries implemented as an Android service [3]. The Java one acts as the client, and the native one acts as the server. These two apps communicate with each other via Android IPC. In this way, the security of native libraries is controlled by Android's existing process isolation security mechanisms, which means the third-party native libraries can not access to the entire application address space. To support arbitrary applications, where source code is not always available, we craft the application separation process in bytecode instead of source code.

To control the third-party native libraries' access to private data and dangerous operations, NativeProtector inserts hooking libraries into the server app which contains the third-party libraries. These hooking libraries intercept the interactions between the third-party native libraries and the system to enforce various security policies. To control the third-party native libraries' access to private data and dangerous operations by JNI calls, we also intercept the JNI calls called by the native code to enforce security policies.

Use Case: For each app to be installed, NativeProtector separates it into two standalone apps. They are both installed on the user's cellphone and the server app is installed as an Android service. When the client app is launched, it starts the server app and binds to the service of the server app. When the client app needs to call the functions in the native libraries, the client app will interact with the server app through IPC.

4 Implementation

We have implemented a prototype of NativeProtector in Java and C programming languages.

4.1 Apk Repackaging

Each Android application is distributed as a package file format which is named APK (Android Application Package). An APK file contains a manifest file named AndroidManifest.xml, the application's code in form of dex bytecode, XML resources like activity layouts, other resources such as images, and the native libraries which are standalone Linux shared object files (.so). To support arbitrary applications, all we worked is on the APK file.

An APK file is actually a ZIP compression package, which can be easily decompressed with decompression tools. Because the Android SDK puts all the application's compiled bytecode in a single file called classes.dex, and the XML files are also compressed, we can not edit the bytecode or XML files directly to add our protection code. We need to take the original APK files, disassemble it to a collection of individual classes and XML files, add NativeProtector's code in them, and then reassemble all the things back to new APK files.

To perform this task we choose *apktool* [5], a tool for Android applications reverse engineering which can decode resources to nearly original form and rebuild them after making some modifications. In NativeProtector *apktool* is taken in the repackaging process. We first take *apktool* to disassemble the APK file to manifest file and resources, the native libraries, and the application's bytecode which is in the *smali* format. Then we use these files to generate the client app source files and server app source files. Meanwhile, we insert our stub libraries into the server app source files to intercept the private data access and dangerous operation for enforcing security policies. Finally, we use *apktool* to reassemble the client app and server app to APK files, and install them on the user's phone.

4.2 Native Libraries Isolation

NativeProtector isolates the native libraries to another application as an Android service. Actually, it generates several AIDL(Android Interface Definition Language [1]) interfaces and assistant smali code, and then modifies the launcher Activity and the JNI calls in the applications to achieve isolation. Figure 2. shows the detailed process of the isolation of NativeProtector. Next, we illustrate some key points in the isolation process.

Generate Server Application. After disassembling the original app's APK file by *apktool*, all smali files of the application are analyzed to record native function information. Then for each native function, NativeProtector creates a corresponding AIDL interface and an Android wrapper function in the server for

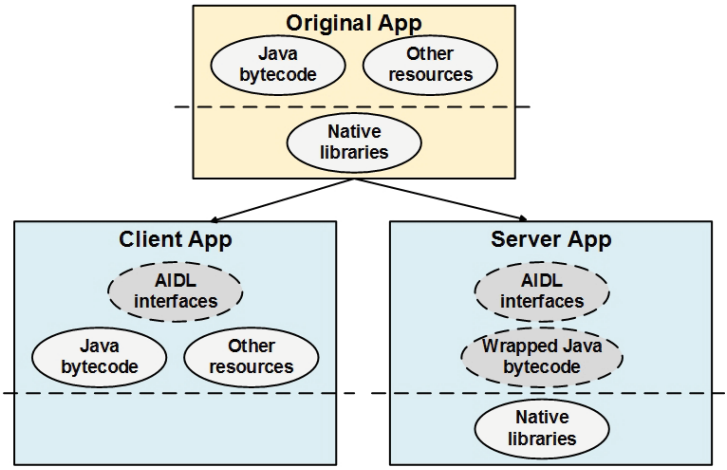


Fig. 2. Isolation process of NativeProtector

communicating with the client. When the client calls the native function, there is an IPC call to the corresponding wrapper function with the same parameter, and the wrapper function invokes the native function through the AIDL interface. Finally, the native functions with AIDL interfaces and wrapper functions are reassembled into the server app using *apktool*.

Generate Client Application. After the server app is generated, we use the generated AIDL interfaces to craft the client app. We analyze all the smali files of the original app automatically to change the native function calls to corresponding AIDL calls. Then we use *apktool* to reassemble the Java bytecode of the original app which has been modified as above mentioned, the AIDL interfaces corresponding to the server app, and the other resource files of the original app to the client app as an APK file.

Modify the Launcher Activity. To make the client app works normally we need to ensure that at the very beginning in the client app, the server app's service is bound to the client app and the client app can call the native functions normally. Launcher activity is the starting execution point of an Android application. A callback method of the activity named onCreate is automatically invoked by the system when the application is launched. So NativeProtector locates this callback method in the launcher activity, and inserts code to bind to the server app to ensure that the client app is launched together with the server app.

4.3 Native Libraries Interception

In the process of the native libraries isolation, NativeProtector instruments the native libraries for intercepting the private data access and dangerous operations. This interception is implemented as some hooking libraries which are packed in the server app, and we let the server app load the hooking libraries and perform the hooking operation for native libraries interception. NativeProtector also provides a policy manager employing various security policies to enforce security. Next we detail the native libraries interception process of NativeProtector.

Efficient Interception. The PLT/GOT structure is used for dynamic linking in Android. The call sites to external functions in Android native libraries are jump instructions to stub functions in PLT, and the stub functions then perform memory load on the entries in the GOT to retrieve the real address of these functions. We can exploit this mechanism to perform the required interposition. We scan every loaded native library and overwrite each GOT entry with pointer to our stub function. In this way, we can intercept the private data access and dangerous operations.

Policies. As we are able to insert hooking libraries which can intercept the private data access and dangerous operations to monitor the third-party native libraries, we can introduce the various security policies in NativeProtector to enforce security. At present, we implement some typical security policies in NativeProtector. In theory, NativeProtector can perform as a flexible framework to adopt many more useful security policies.

In NativeProtector, we monitor mainly three kinds of operations, including accessing private data, connecting to remote server and sending SMS (Simple Message Service) messages. We consider those operations, because they cover the two ends of path that may leak private data. The policies taken by NativeProtector to regulate the third-party native libraries in these three operations are described below:

- **Private Data Policy.** This policy protects the user's private data such as IMEI, IMSI, phone number, location, stored SMS messages and contact list. These private data can be accessed from the system services provided by Android Framework APIs. These Android Framework APIs call a single call to the *ioctl()* function. NativeProtector intercepts the calls to *ioctl()*, and parses the data passed in the calls to determine which service is accessed. Then we can know which private data is accessed by the native libraries and decide whether allow the private data access of the native libraries.
- **Network Policy.** Android uses *connect()* function for socket connection. All Internet connections in Android call this function eventually. NativeProtector can intercept this function to control the Internet access of the native libraries. We restrict the native libraries to connect to only a specific set of IP addresses and prevent the native libraries from connecting to malicious IP addresses.

With these whitelist and blacklist policies, we can ensure the native libraries are regulated int network, and the normal use of the native libraries is not affected. The whitelist and blacklist can be managed by user, or be retrieved from a trusted server.

– **SMS Policy.** In Android framework, applications can not send SMS messages on their own. The applications must invoke RPCs to the SMS service through Binder. All the interactions with Binder call *ioctl()* function eventually. As we mentioned above, NativeProtector intercepts the calls to *ioctl()*. Thus we can get the destination number and the content of the SMS message to inform the users, and decide whether to allow this operation or not. Meanwhile we can take a blacklist policy to prevent the native libraries from sending SMS messages to premium numbers. As NativeProtector only block sending SMS message to premium numbers, calling to those numbers is not blocked unless the phone call policy is adopted.

5 Evaluation

In this section, we evaluate the effectiveness, compatibility and performance of our prototype of NativeProtector. The experiments are performed on a Google Nexus 4 phone running Android 4.4.2.

5.1 Effectiveness

To show the effectiveness of NativeProtector, we have manually designed a demo app. This demo app contains a malicious but inseparable native library, which is used by the demo app for network connection. But this malicious native library gets the location information of the phone and send to a known malicious server. The demo app uses native system calls to perform this process instead of Java APIs.

This demo app demonstrates that the third-party native libraries may make use of the permissions assigned to the applications and cause security violations. As for NativeGuard [21], it simply separates the native libraries to another server app, and gives no permissions to the server app. This approach indeed restricts the third-party native libraries' access to all the application's process address which can change the execution of java code, but it also leads to the result that the third-party native libraries can not work any more. In this app, the native library can not connect to the network, even for the benign servers.

NativeProtector can improve the situation by combining separation and interception. For this demo app, NativeProtector separates the native library to the server app and generates hooking libraries to the server app. The server app has the permissions same as the original app, so the third-party native library can work well. But when the third-party native library accesses to the phone location (private data in the test), and when the third-party native library connects to the malicious IPs in the blacklist, the hooking libraries will intercept these

dangerous function calls and block them. In this way, NativeProtector can help the app to use the third-party native libraries functionality and prevent applications from the malicious third-party native libraries.

5.2 Compatibility

To test the compatibility of our NativeProtector prototype, we have collected 20 popular apps from APKPure [4] store, a applications store which can ensure the applications download from it are the same as Google Play store. These applications are downloaded from the leaderboard of the hot free applications chart of the store in November 2015. NativeProtector successfully separates 15 of 20 applications. The five failed applications are due to the *apktool*. Two of them fail in the disassembling stage before NativeProtector's separation, and three of them fail in the assembling stage after NativeProtector's separation. Then we manually test the applications after separation by playing with the instrumented applications and using the functionalities of the instrumented applications. In the test, all the 15 applications processed by NativeProtector can work well. During the testing, we found that there are response delays in some applications. But on the whole, NativeProtector introduces acceptable overhead and does not affect the app's functionalities. Details about the evaluated applications are presented in Table 1.

5.3 Performance

For NativeProtector, a security framework which takes process isolation and system call interception, the runtime overhead depends greatly on the intensity of context switches and API invocations we intercepted (like *connect()*). In Sect. 5.2, not mach delay is felt when testing in the real-world applications hardened by NativeProtector. To quantify the performance overhead, we talk about the NativeProtector's performance overhead in the extreme cases. We crafted two artificial application to represent the two extreme cases. One is testing the influence of the intensity of context switches on the NativeProtector's performance overhead, and the other is testing the NativeProtector's overhead affected by the API invocations we intercepted.

Firstly, we crafted an application which compresses a medium size file (about 5.6 MB) stored on the phone with the popular Zlib library. The Java code of the application divides this file to small segments and passes one to Zlib which perform the compression. Then the Java code passes the next one to Zlib. When Zlib library is sandboxed by NativeProtector, the segments size has a great impact on the performance overhead, as small segment size means frequent context switches between Java and native code. Table 2 presents the overhead introduced by enabling NativeProtector in Zlib benchmark application that compresses the file with different segment size. All the performance time is the average over 5 tests. The results table illustrates that higher overhead comes up with smaller segments size. And the overhead can be 81.79 % when the segments size is 1KB. But in the real world, few applications will perform context switches as frequently

Table 1. Apps used in compatibility test (∗: failed by *apktool*)

App	Size	Package name
DuoLingGo	9 M	com.duolingo
ES File Browser∗	5.9 M	com.estrongs.android.pop
Shadowsocks	3.7 M	com.github.shadowsocks
IMO	6.7 M	com.imo.android.imoim
Arrow Desktop	4.5 M	com.microsoft.launcher
Microsoft Outlook	18.1 M	com.microsoft.office.outlook
Snapseed	20.7 M	com.niksoftware.snapseed
Brightest LED Flashlight	5.1 M	com.surpax.ledflashlight.panel
Tiger VPN	5.8 M	com.tigervpns.android
Tumblr	21.1 M	com.tumblr
Twitter	21.7 M	com.twitter.android
HideMe VPN	5.8 M	io.hideme.android
OpenVPN Connect	2.3 M	net.openvpn.openvpn
New York Times∗	1.2 M	org.greatfire.nyt
WiFi key∗	6.3 M	com.halo.wifikey.wifilocating
Touch VPN	4.7 M	com.northghost.touchvpn
Firefox	40.1 M	org.mozilla.firefox
AliExpress Shopping App∗	10 M	com.alibaba.aliexpresshd
Clean Master	16.6 M	com.cleanmaster.mguard
WiFi Toolbox∗	7.7 M	mobi.wifi.toolbox

as this test. They utilize native libraries to perform CPU-intensive operations like image compression and conversion, without frequently making context switches. Thus NativeProtector brings acceptable overhead to the real-world applications.

Secondly, we crafted an application which use native libraries to connect to a remote server and send a message. This application tries to connect the remote sever and send message ten times. After these operations are done, it logs the time consumed. We tested both the original application and the application hardened by NativeProtector for five times against the randomness. Table 3 shows the result of this test. We can see that NativeProtector introduces an overhead of 50 % in this test, which we believe is acceptable as real-world applications will not invoke the APIs NativeProtector intercepted frequently.

In summary, the evaluation demonstrates that NativeProtector can enforce security on the native libraries in Android applications with acceptable overhead in most real-world applications where context switches between Java and native code are not frequent and the API invocations we intercepted are not frequently used.

Table 2. Performance on Zlib benchmark application

Segments Size	Without NativeProtector	With NativeProtector	Overhead
1 KB	21905 ms	39822 ms	81.79 %
2 KB	15093 ms	23948 ms	58.67 %
4 KB	10585 ms	15507 ms	46.50 %
8 KB	8993 ms	11399 ms	26.75 %
16 KB	8494 ms	9585 ms	12.84 %

Table 3. Performance on API invocations

	First test	Second test	Third test	Forth test	Fifth test	Average
Without NP	608 ms	543 ms	512 ms	438 ms	619 ms	544 ms
With NP	960 ms	743 ms	788 ms	672 ms	916 ms	816 ms
Overhead	50 %					

6 Related Work

6.1 Android App Security

With the increasing popularity of Android and the increasing malware threats, many approaches to secure Android applications have been proposed. Some of them extend the Android framework to perform fine-grained control of Android applications at runtime [10,11,14,18,22,28]. AppFence [14] retrofits the Android operating system to return mock data of the sensitive resources to the imperious applications. CRePE [11] allows both the user and authorized third parties to enforce fine-grained context-related polices. Deepdroid [22] intercepts Binder transactions and traces system calls to provide portability and fine-grained control. The other approaches perform security enforcement at the application layer [8,9,12,17,23,27]. Aurasium [23] uses native library interposing to enforce arbitrary policies at runtime. Appcage [27] hooks into the application's Dalvik virtual machine instance to sandbox the application's dex code and uses SFI (Software Fault Isolation) to sandbox the application's native code. Boxify [9] combines the strong security guarantees of OS security extensions with the deployability of application layer solutions by building on isolated processes to restrict privileges of untrusted applications and introducing a novel application virtualization environment. NativeProtector takes the application layer approach. It can run in the user's phone and does not need to modify the Android framework, so it can be easily deployed.

6.2 Untruseted Code Isolation

Android's UID-based sandboxing mechanism strictly isolates different applications in different processes. With this strong security boundary naturally supported by Android, many approaches have been proposed to isolate untrusted code

in another process [15,19,21,24]. Dr. Android and Mr. Hide [15] revokes all Android platform permissions from the untrusted applications and applies code rewriting techniques to replace well-known security-sensitive Android API calls in the monitored application with calls to the separate reference monitor application that acts as a proxy to the application framework. AdSplit [19] and AFrame [24] isolate the third-party advertising libraries which is the JAR format into separate processes. NativeGuard [21] isolates native libraries into a non-privileged application. But the benign native libraries which need permission to perform legal task can not work any more, because it lacks the fine-grained access control of the native libraries. NativeProtector adopts the idea of isolation, and instruments the native libraries to take fine-grained access control of native libraries.

7 Conclusion

We have presented the design, implementation and the evaluation of NativeProtector, a system to regulate the third-party native libraries in Android applications. NativeProtector separates the native libraries and generated hooking libraries to another server app, and the rest part of the original app is generated as the client app. The client app binds to the server app at the launching time, and all native function calls are replaced with the IPCs to the server app. The hooking libraries intercept system calls in server app to enforce security of native libraries in the server app. We have implemented a prototype of NativeProtector. Our evaluation shows that NativeProtector can successfully regulate the third-party native libraries in Android applications and introduces acceptable overhead.

References

1. Android Interface Definition Language (AIDL). http://developer.android.com/intl/zh-cn/guide/components/aidl.html
2. Android NDK. http://developer.android.com/intl/zh-cn/ndk/index.html
3. Android Service. http://developer.android.com/reference/android/app/Service.html
4. apkpure.com. https://apkpure.com
5. Apktool. http://ibotpeaches.github.io/Apktool/
6. Android market share in Q2 (2015). http://www.idc.com/prodserv/smartphone-os-market-share.jsp
7. PLT and GOT, the key to code sharing and dynamic libraries. https://www.technovelty.org/linux/plt-and-got-the-key-to-code-sharing-and-dynamic-libraries.html
8. Backes, M., Gerling, S., Hammer, C., Maffei, M., von Styp-Rekowsky, P.: AppGuard – Enforcing user requirements on android apps. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013 (ETAPS 2013). LNCS, vol. 7795, pp. 543–548. Springer, Heidelberg (2013)
9. Backes, M., Bugiel, S., Hammer, C., et al.: Boxify: Full-fledged app sandboxing for stock Android. In: Proceedings of 24th USENIX Security Symposium (USENIX Security 2015) (2015)

10. Bugiel, S., Heuser, S., Sadeghi, A.R.: Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: Proceedings of 22th USENIX Security Symposium (USENIX Security 2013), pp. 131–146 (2013)
11. Conti, M., Nguyen, V.T.N., Crispo, B.: CRePE: Context-related policy enforcement for Android. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 331–345. Springer, Heidelberg (2011)
12. Davis, B., Sanders, B., Khodaverdian, A., et al.: I-ARM-Droid: A rewriting framework for in-app reference monitors for android applications. In: Proceedings of Mobile Security Technologies (2012)
13. Enck, W., Gilbert, P., Han, S., et al.: TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. Proc. ACM Trans. Comput. Syst. (TOCS) **32**(2), 5 (2014)
14. Hornyack, P., Han, S., Jung, J., et al.: These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 639–652. ACM (2011)
15. Jeon, J., Micinski, K.K., Vaughan, J.A., et al.: Dr. Android and Mr. Hide: fine-grained permissions in android applications. In: Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 3–14. ACM (2012)
16. Nauman, M., Khan, S., Zhang, X.: Apex: extending android permission model and enforcement with user-defined runtime constraints. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 328–332. ACM (2010)
17. Rasthofer, S., Arzt, S., Lovat, E., et al.: Droidforce: enforcing complex, data-centric, system-wide policies in android. In: 2014 Ninth International Conference on Availability, Reliability and Security (ARES), pp. 40–49. IEEE (2014)
18. Russello, G., Jimenez, A.B., Naderi, H., et al.: Firedroid: hardening security in almost-stock android. In: Proceedings of the 29th Annual Computer Security Applications Conference, pp. 319–328. ACM (2013)
19. Shekhar, S., Dietz, M., Wallach, D.S: AdSplit: Separating smartphone advertising from applications. In: Proceedings of 21th USENIX Security Symposium, pp. 553–567 (2012)
20. Siefers, J., Tan, G., Morrisett, G.: Robusta: Taming the native beast of the JVM. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 201–211. ACM (2010)
21. Sun, M., Tan, G.: NativeGuard: Protecting android applications from third-party native libraries. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks, pp. 165–176. ACM (2014)
22. Wang, X., Sun, K., Wang, Y., et al.: DeepDroid: dynamically enforcing enterprise policy on android devices. In: Proceedings of 22nd Annual Network and Distributed System Security Symposium (NDSS 2015). The Internet Society (2015)
23. Xu, R., Sadi, H., Anderson, R.: Aurasium: Practical policy enforcement for android applications. In: Proceedings of 21th USENIX Security Symposium, pp. 539–552 (2012)
24. Zhang, X., Ahlawat, A., Du, W.: AFrame: isolating advertisements from mobile applications in Android. In: Proceedings of the 29th Annual Computer Security Applications Conference, pp. 9–18. ACM (2013)
25. Zhang, Y., Yang, M., Xu, B., et al.: Vetting undesirable behaviors in android apps with permission use analysis. In: Proceedings of the ACM SIGSAC Conference on Computer & Communications Security, pp. 611–622. ACM (2013)

26. Zhao, Z., Osono, F.C.C.: TrustDroid: Preventing the use of SmartPhones for information leaking in corporate networks through the used of static analysis taint tracking. In: MALWARE, pp. 135–143 (2012)
27. Zhou, Y., Patel, K., Wu, L., et al.: Hybrid user-level sandboxing of third-party android apps. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (2015)
28. Zhou, Y., Zhang, X., Jiang, X., Freeh, V.W.: Taming information-stealing smartphone applications (on Android). In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 93–107. Springer, Heidelberg (2011)

A Progress-Sensitive Flow-Sensitive Inlined Information-Flow Control Monitor

Andrew Bedford[1(✉)], Stephen Chong[2], Josée Desharnais[1], and Nadia Tawbi[1]

[1] Laval University, Quebec, Canada
andrew.bedford.1@ulaval.ca
[2] Harvard University, Cambridge, USA

Abstract. We present a novel progress-sensitive, flow-sensitive hybrid information-flow control monitor for an imperative interactive language. Progress-sensitive information-flow control is a strong information security guarantee which ensures that a program's progress (or lack of) does not leak information. Flow-sensitivity means that this strong security guarantee is enforced fairly precisely: we track information flow according to the source of information and not to an a priori given variable security level. We illustrate our approach on an imperative interactive language. Our hybrid monitor is inlined: source programs are translated, by a type-based analysis, into a target language that supports dynamic security levels. A key benefit of this is that the resulting monitored program is amenable to standard optimization techniques such as partial evaluation.

1 Introduction

Information-flow control is a promising approach to enable trusted systems to interact with untrusted parties, providing fine-grained application-specific control of confidential and untrusted information. Static mechanisms for information-flow control (such as security type systems [12,14]) analyse a program before execution to determine whether its execution satisfies the information flow requirements. This has low runtime overhead, but can generate many false positives. Dynamic mechanisms (e.g., [4]) accept or reject individual executions at runtime and thus can incur significant runtime overheads. *Hybrid information-flow control* techniques (e.g., [8]) combine static and dynamic program analysis and strive to achieve the benefits of both: precise (i.e., per-execution) enforcement of security and low runtime overhead.

We present a novel progress-sensitive [2], flow-sensitive hybrid information-flow control monitor for an imperative interactive language. Our monitor prevents leaks of confidential information, notably via *progress channels*, while limiting over approximation, thanks to *flow sensitivity* and its inline nature. Our monitor is inlined: source programs are translated into a target language that supports dynamic security levels [15]. The type-based translation inserts commands to track the security levels of program variables and contexts, and to

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 352–366, 2016.
DOI: 10.1007/978-3-319-33630-5_24

control information flow. A key benefit is that the resulting monitored program is amenable to standard optimization techniques such as partial evaluation [7].

The translation to the target language performs a static analysis using three security levels: L (for low-security information), H (for high-security information), and U (for unknown information). If the program is statically determined to be insecure, then it is rejected. Otherwise, the translation of the program dynamically tracks the unknown security levels, and ensures that no leak occurs.

Our main contributions are twofold. This work is one of the first hybrid monitor that enforces both flow and progress-sensitive information security; moreover, the combination of channel-valued variables, flow-sensitivity and progress-sensitivity presents a couple of issues that we solve.

Motivating Examples

Channel Variables. Our source language supports channel variables whose security level can be statically unknown. This leads to use a special security level, U, which delays the decision to accept or reject certain programs to runtime. Indeed, a channel level needs upward or downward approximation according to its use and this cannot be approximated, as the following example shows.

```
    if lowValue  > 0
      then  d := lowChannel
      else  d := highChannel end;
(* Case 1  *)                    (* Case 2 *)
send highValue to d              x := read d;
(*to be rejected if d is L*)     send x to lowChannel
                                 (*to be rejected if d is H*)
```

Listing 1.1: We cannot be pessimistic about channel variables

Progress Channels. The progress of a program, observable through its outputs, can reveal information. In the following program, the occurrence of an output on the public channel reveals a confidential information controlling the loop termination.

```
while highValue > 0
  do skip end;
send 42 to lowChannel
```

Listing 1.2: Progress leak

The most common way to prevent leaks through progress channels is to forbid loops whose execution depends on confidential information [10,13], but it leads to the rejection of many secure programs, such as the following.

```
while highValue > 0 do
  highValue := highValue - 1 end;
send 42 to lowChannel
```

Listing 1.3: Loop that always terminates

Inspired by Moore et al. [9], we use an oracle to determine the termination behaviour of loops. If it tells that a loop always terminates (cf Listing 1.3), then there is no possible leak of information. If the oracle says it may diverge, then a risk of information leak is flagged. The oracle is a parameter based on termination analysis methods brought from the literature [6].

Structure. In Sect. 2, we present the imperative language used to illustrate our approach. Section 3 defines the non-interference property. Section 4 describes our typed-based instrumentation mechanism, explains the type system, and presents the target language in which the instrumented programs are written; it is an extension of the source language with dynamic security levels. Section 5 is a summary of related work. Finally, we conclude in Sect. 6.

2 Source Language

Source programs are written in a simple imperative language. We suppose that the interaction of a program with its environment is done through channels. Channels can be, for example, files, users, network channels, keyboards, etc. These channel constants are associated to a priori security levels, private or public. This is more realistic than requiring someone to manually define the level of every variable of the program; their level can instead be inferred according to the sources of information they may hold.

2.1 Syntax

Let $\mathcal{V}$ be a set of identifiers for variables, and $\mathcal{C}$ a set of predefined communication channels. The syntax is as follows.

$$
\begin{array}{lrcl}
\textit{(variables)} & x & \in & \mathcal{V} \cup \mathcal{C} \\
\textit{(integer constants)} & n & \in & \mathbb{Z} \\
\textit{(expressions)} & e & ::= & x \mid n \mid e_1 \ \mathbf{op}\ e_2 \mid \mathbf{read}\ x \\
\textit{(commands)} & \mathit{cmd} & ::= & \mathbf{skip} \mid x := e \mid \mathbf{if}\ e\ \mathbf{then}\ \mathit{cmd}_1\ \mathbf{else}\ \mathit{cmd}_2\ \mathbf{end} \mid \\
 & & & \mathbf{while}\ e\ \mathbf{do}\ \mathit{cmd}\ \mathbf{end} \quad \mathit{cmd}_1; \mathit{cmd}_2 \quad \mathbf{send}\ x_1\ \mathbf{to}\ x_2
\end{array}
$$

Values are integers (we use zero for false and nonzero for true), or channel names. Symbol **op** stands for arithmetic or logic binary operators. We write *Exp* for the set of expressions. W.l.o.g., we assume each channel consists of one value, which can be read or modified through **read** operation and **send** command respectively. It is easy to generalize to channels consisting in sequences of values.

2.2 Semantics

A memory $m : \mathcal{V} \uplus \mathcal{C} \to \mathbb{Z} \uplus \mathcal{C}$ is a partial map from variables and channels to values, where the value of a channel is the last value sent to this channel. More precisely a memory is the disjoint union of two maps of the following form:

$$m_v : \mathcal{V} \to \mathbb{Z} \uplus \mathcal{C}, \qquad m_c : \mathcal{C} \to \mathbb{Z},$$

where $\uplus$ stands for the disjoint union operator. We omit the subscript whenever the context is clear. We write $m(e) = r$ to indicate that the evaluation of expression e under memory m returns r.

The semantics of the source language is mostly standard and is illustrated in Fig. 1. Program configurations are tuples $\langle cmd, m, o\rangle$ where cmd is the command to be evaluated, m is the current memory and o is the current output trace. A transition between two configurations is denoted by the $\longrightarrow$ symbol. We write $\longrightarrow^*$ for the reflexive transitive closure of the $\longrightarrow$ relation.

We write $v :: vs$ for sequences where v is the first element of the sequence, and vs is the rest of the sequence. We write ϵ for the empty sequence. An output trace is a sequence of output events: it is of the form $o = (v_0, ch_0) :: (v_1, ch_1) :: \ldots$ where $v_k \in \mathbb{Z}$ is an integer value, and ch_k is a channel, $k \in \mathbb{N}$. The rule for sending a value appends a new output event to the end of the trace. (We abuse notation and write $o :: (v, ch)$ to indicate event (v, ch) appended to trace o.)

$$\text{(Skip)}\quad \langle \mathbf{skip}, m, o\rangle \longrightarrow \langle \mathbf{stop}, m, o\rangle \qquad \text{(Assign)}\quad \frac{m(e) = r}{\langle x := e, m, o\rangle \longrightarrow \langle \mathbf{stop}, m[x \mapsto r], o\rangle}$$

$$\text{(Send)}\quad \frac{m(x_1) = v \in \mathbb{Z} \qquad m(x_2) = ch \in \mathcal{C}}{\langle \mathbf{send}\ x_1\ \mathbf{to}\ x_2, m, o\rangle \longrightarrow \langle \mathbf{stop}, m[ch \mapsto v], o :: (v, ch))\rangle}$$

$$\text{(Seq1)}\quad \frac{\langle cmd_1, m, o\rangle \longrightarrow \langle \mathbf{stop}, m', o'\rangle}{\langle cmd_1; cmd_2, m, o\rangle \longrightarrow \langle cmd_2, m', o'\rangle} \qquad \text{(Seq2)}\quad \frac{\langle cmd_1, m, o\rangle \longrightarrow \langle cmd_1', m', o'\rangle \qquad cmd_1' \neq \mathbf{stop}}{\langle cmd_1; cmd_2, m, o\rangle \longrightarrow \langle cmd_1'; cmd_2, m', o'\rangle}$$

$$\text{(If)}\quad \frac{m(e) \neq 0 \Longrightarrow i = 1 \qquad m(e) = 0 \Longrightarrow i = 2}{\langle \mathbf{if}\ e\ \mathbf{then}\ cmd_1\ \mathbf{else}\ cmd_2\ \mathbf{end}, m, o\rangle \longrightarrow \langle cmd_i, m, o\rangle}$$

$$\text{(Loop1)}\quad \frac{m(e) \neq 0}{\langle \mathbf{while}\, e\, \mathbf{do}\, cmd\, \mathbf{end}, m, o\rangle \longrightarrow \langle cmd; \mathbf{while}\, e\, \mathbf{do}\, cmd\, \mathbf{end}, m, o\rangle}$$

$$\text{(Loop2)}\quad \frac{m(e) = 0}{\langle \mathbf{while}\ e\ \mathbf{do}\ cmd\ \mathbf{end}, m, o\rangle \longrightarrow \langle \mathbf{stop}, m, o\rangle}$$

Fig. 1. Semantics of the source language

We write $\langle cmd, m, \epsilon\rangle \downarrow o$ if execution of configuration $\langle cmd, m, \epsilon\rangle$ can produce trace o, where o may be finite or infinite. For finite o, $\langle cmd, m, \epsilon\rangle \downarrow o$ holds if there is a configuration $\langle cmd', m', o\rangle$ such that $\langle cmd, m, \epsilon\rangle \longrightarrow^* \langle cmd', m', o\rangle$. For infinite o, $\langle cmd, m, \epsilon\rangle \downarrow o$ holds if for all traces o' such that o' is a finite prefix of o, we have $\langle cmd, m, \epsilon\rangle \downarrow o'$.

3 Security

We define an execution as secure if the outputs on public channels do not reveal any information about the inputs of private channels. This is a standard form of

non-interference (e.g., [12,14]) adapted to our particular language model. More formally, we require that any two executions of the programs starting from initial memories that have the same public channel inputs, produce the same publicly observable outputs. This means that an observer of the public output could not distinguish the two executions, and thus learns nothing about the inputs of private channels.

Before formally defining non-interference, we first introduce some helpful technical concepts. We assume a lattice of security levels $(\mathcal{L}, \sqsubseteq)$ with two elements: L (Low) for public information and H (High) for private information, ordered as $L \sqsubseteq H$. The *projection of trace o to security level ℓ*, written $o \restriction \ell$, is its restriction to output events whose channels' security levels are less than or equal to ℓ. Formally,

$$\epsilon \restriction \ell = \epsilon$$

$$((v, ch) :: o) \restriction \ell = \begin{cases} (v, ch) :: (o \restriction \ell) & \text{if } \mathit{levelOfChan}(ch) \sqsubseteq \ell \\ o \restriction \ell & \text{otherwise} \end{cases}$$

where $\mathit{levelOfChan}(ch)$ denotes the security level of channel ch (typically specified by the administrator).

We say that two memories m and m' *differ only on private channel inputs* if $m_v = m'_v$ and

$$\forall ch \in \mathcal{C}.\mathit{levelOfChan}(ch) = L \Rightarrow m_c(ch) = m'_c(ch).$$

Definition 1 (Progress-Sensitive Non-Interference).
We say that a program p satisfies progress-sensitive non-interference *if for any two memories m and m' that agree on public variables and public channel inputs, and for any (finite or infinite) trace o such that $\langle p, m, \epsilon\rangle \downarrow o$, then there is some trace o', such that $\langle p, m', \epsilon\rangle \downarrow o'$ and $o \restriction L = o' \restriction L$.*

This definition of non-interference is progress-sensitive in that it assumes that an observer can distinguish an execution that will not produce any additional observable output (due to termination or divergence) from an execution that will make progress and produce additional observable output. Progress-insensitive definitions of non-interference typically weaken the requirement that $o \restriction L = o' \restriction L$ to instead require that $o \restriction L$ is a prefix of $o' \restriction L$, or vice versa.

4 Type-based Instrumentation

We enforce non-interference by translating source programs to a target language that enables the program to track the security levels of its variables. The translation performs a type-based static analysis of the source program, and rejects programs that clearly leak information (i.e. the translation fails).

In this section, we first present the security types for the source language (in order to provide intuition for the type-directed translation) followed by the description of the target language, which extends the source language with runtime representation of security levels. We then present the translation from the source language to the target language.

4.1 Source Language Types

Source language types are defined according to the following grammar. The security types are defined as follows:

$$\begin{array}{ll} \textit{(security levels, } \mathcal{L}\textit{)} & \ell ::= \ L \mid U \mid H \\ \textit{(value types, } ValT\textit{)} & \sigma ::= \mathit{int} \mid \mathit{int}_\ell \ \mathit{chan} \\ \textit{(variable types, } VarT\textit{)} & \tau ::= \sigma_\ell \end{array}$$

Security levels in types include L and H, and also U (Unknown), which is used to represent a statically unknown security level. The translated program will explicitly track these statically unknown security levels at runtime. The security levels are organized in a lattice $(\mathcal{L}, \sqsubseteq)$, where $\mathcal{L} = \{L, U, H\}$ and $L \sqsubseteq U \sqsubseteq H$, $(H \not\sqsubseteq U \not\sqsubseteq L)$. The associated supremum is denoted $\sqcup$. We derive two order relations that allow us to deal with the uncertainty level.

Definition 2. *The relations* $\sqsubseteq_s$, surely less than, *and* $\sqsubseteq_m$, maybe less than, *are defined as follows*

$$\begin{array}{ll} \ell_1 \sqsubseteq_s \ell_2 & \textit{if } (\ell_1 \sqsubseteq \ell_2) \wedge \neg(\ell_1 = \ell_2 = U) \\ \ell_1 \sqsubseteq_m \ell_2 & \textit{if } (\ell_1 \sqsubseteq \ell_2 \vee \ell_1 = U \vee \ell_2 = U) \end{array}$$

Intuitively, we have $\ell \sqsubseteq_s \ell'$ when we can be sure statically that $\ell \sqsubseteq \ell'$ will be true at runtime, and we have $\ell \sqsubseteq_m \ell'$ when it is possible that $\ell \sqsubseteq \ell'$ at runtime. For example, $U \not\sqsubseteq_s L$ but $U \sqsubseteq_m L$.

Value types are the types of integers (*int*) and channels. Type $\mathit{int}_\ell \ \mathit{chan}$ is the type of a channel whose values are of security level ℓ.

Variables types associate a security level with a value type. Intuitively, σ_ℓ represents the type of a variable whose value type is σ, and whose variable type is ℓ, the latter is an upper bound of the information level influencing the value.

We instrument source programs to track at runtime the security levels that are statically unknown. That is, if a variable x has type σ_U for some value type σ, then the instrumented program will have a variable that explicitly tracks the security level of variable x. Moreover, if σ is the unknown channel type $(\mathit{int}_U \ \mathit{chan})$ then the instrumented program will have a variable that explicitly tracks the security level of the channel that is assigned to x. In order to track these security levels, our target language allows their runtime representation.

The Uncertain Level. As illustrated in Listing 1.1, a channel level needs upward or downward approximation according to its use. This is the main reason underlying the use of the uncertainty level U. After the conditionals of that listing, `d` has type $(\mathit{int}_U \mathit{chan})_L$ because it contains either a low or high channel and its value is assigned in a context of level L. Our typing system accepts this program in both `Case 1` and `Case 2`, but inserts runtime checks. If the condition `lowValue` $>$ `0` is false at runtime, then sending of a `highValue` on `d` would be safe, and `Case 1` should be accepted, while `Case 2` should be rejected since it attempts to send a high level value to a public channel. On the contrary, if `lowValue` $>$ `0` appears to be false at runtime, then `Case 1` should be accepted and `Case 2` rejected.

The uncertainty is unavoidable in the presence of flow sensitivity and channel variables. Indeed, we point out that we cannot be pessimistic about the level of variable channels in this program. The output command suggests that a safe (yet too strong) approximation for `d` would be a low security level. Yet, the input command suggests that a safe (yet too strong) approximation for `d` would be a *high* security level, which contradicts the previous observation. Consequently, if we are to accept the program in Listing 1.1, in both cases, we need an alternative security type, U, to carry on with the analysis.

4.2 Syntax and Semantics of Target Language

Our target language is inspired by the work of Zheng and Myers [15], which introduced a language with first-class security levels, and a type system that soundly enforces non-interference in this language. The syntax of our target language is defined as follows. The main difference with the source language is that it adds support for *level variables* (regrouped in the set $\mathcal{V}_{level}$), a runtime representation of security levels.

$$
\begin{array}{lrcl}
\textit{(variables)} & x & \in & \mathcal{V} \cup \mathcal{C} \\
\textit{(level variables)} & \tilde{x} & \in & \mathcal{V}_{level} \\
\textit{(integer constants)} & n & \in & \mathbb{Z} \\
\textit{(basic levels)} & k & ::= & L \mid H \\
\textit{(level expressions)} & \ell & ::= & k \mid \tilde{x} \mid \overline{\ell} \mid \ell_1 \sqcup \ell_2 \mid \ell_1 \sqcap \ell_2 \\
\textit{(integer expressions)} & \mathit{exp} & ::= & x \mid n \mid \mathit{exp}_1 \ \mathbf{op} \ \mathit{exp}_2 \mid \mathbf{read}\, x \\
\textit{(expressions)} & e & ::= & \mathit{exp} \mid \ell \\
\textit{(commands)} & \mathit{cmd} & ::= & \mathbf{skip} \mid (x_1, \ldots, x_n) := (e_1, \ldots, e_n) \mid \\
 & & & \mathbf{if}\ e\ \mathbf{then}\ \mathit{cmd}_1\ \mathbf{else}\ \mathit{cmd}_2\ \mathbf{end} \mid \mathit{cmd}_1; \mathit{cmd}_2 \mid \\
 & & & \mathbf{while}\ e\ \mathbf{do}\ \mathit{cmd}\ \mathbf{end} \mid \mathbf{send}\ x_1\ \mathbf{to}\ x_2 \mid \\
 & & & \mathbf{if}\ \ell_1 \sqsubseteq \ell_2\ \mathbf{then}\ (\mathbf{send}\, x_1\, \mathbf{to}\, x_2)\ \mathbf{else\ fail\ end}
\end{array}
$$

Dynamic types will allow a verification of types at runtime: this is the goal of the new send command, nested in a conditional – call it a *guarded send* – that permits to check some conditions on security levels before sending a given variable to a channel. If the check fails, the program aborts. In the target language, only security levels L and H are represented at runtime. The security level U used in the source language typing is replaced by variables and expressions on variables. Level expressions support operators for supremum, infimum and complement (where $\overline{L} = H$ and $\overline{H} = L$); these are defined in Sect. 4.3.

For simplicity, we assume that security levels can be stored only in a restricted set of variables $\mathcal{V}_{level} \subseteq \mathcal{V}$. Thus, the variable part m_v of a memory m now has the following type $m_v : (\mathcal{V}_{level} \to \{L, H\}) \uplus (\mathcal{V} \setminus \mathcal{V}_{level} \to \mathbb{Z} \uplus \mathcal{C})$. Furthermore we assume that $\mathcal{V}_{level}$ contains variables $\_\mathsf{pc}$ and $\_\mathsf{hc}$, and, for each variable $x \in \mathcal{V} \setminus \mathcal{V}_{level}$ there exist level variables x_{lev}; for channel variables, we also have a level variable for their content, that is, the level of the information stored in the channel that the variables point to, written x_{ch}. They will be used in instrumented programs to track security levels. For example, if x is a channel variable of security

type $(int_\ell\ chan)_{\ell'}$, then the values of these variables should be $x_{\mathsf{ch}} = \ell$ and $x_{\mathsf{lev}} = \ell'$ (this will be ensured by our instrumentation). Variables $\_\mathsf{pc}$ and $\_\mathsf{hc}$ hold the security levels of the context and halting context respectively. What these are will be explained in Sect. 4.3. Note that the simultaneous assignment $(x_1, \ldots, x_n) := (e_1, \ldots, e_n)$ is introduced to ensure coherence between the value of a label variable and the level of the value assigned to the corresponding variable. For all other common commands, the semantics of the target language is the same as in the source language.

4.3 Instrumentation as a Type System

Our instrumentation algorithm is specified as a type system in Fig. 2. Its primary goal is to inline monitor actions in the program under analysis, thereby generating a safe version of it. Its secondary goal is to reject programs that contain obvious leaks of information. The inlined actions are essentially updates and checks of level variables to prevent a **send** command from leaking information.

The typing rules of variables and constants have judgements of the form $\Gamma \vdash e : \sigma_\ell$, telling that σ_ℓ is the variable type of e. The instrumentation judgements are of the form $\Gamma, pc, hc \vdash cmd : t, h, \Gamma', [\![cmd]\!]$ where $\Gamma, \Gamma' : \mathcal{V} \uplus \mathcal{C} \to VarT$ are typing environments (initially empty), cmd is the command under analysis, pc is the program context, hc is the halting context, t is the termination type of cmd, h is the updated halting context, and $[\![cmd]\!]$ is the instrumented command. The latter is often presented using a macro whose name starts with *gen*. The program context, pc, is used to keep track of the security level in which a command is executed, in order to detect implicit flows. The halting context, hc, is used to detect progress channels leaks. It represents the level of information that could cause the program to *halt* (due to a failed guarded send command) or diverge (due to an infinite loop). In other words, it is the level of information that could be leaked through progress channels by an output. The termination t of a command is propagated in order to keep the halting context up to date. We distinguish five *termination types* $\mathcal{T} = \{T, D, M_L, M_U, M_H\}$, where T means that a command terminates for all memories, D, diverges for all memories, M_L, M_H and M_U mean that a command's termination is unknown statically; the subscript is used to indicate on which level the termination depends. For example, the termination of the loop in Listing 1.2 is M_H because it can either terminate or diverge at runtime, and this depends on information of level H. The loop in Listing 1.3 on the other hand is of termination type T because, no matter what the value of `highValue` is, it will always eventually terminate. Similarly, a loop whose condition is always true will have termination type D since it always diverges. The precision of this analysis depends on the oracle precision.

The instrumentation of a program p begins by inserting commands to initialize a few level variables: $\_\mathsf{pc}$, $\_\mathsf{hc}$ are initialized to L, as well as the level variables x_{lev} and x_{ch} for each variable $x \in \mathcal{V}$ appearing in p. Similarly, level variables c_{lev} and c_{ch} associated with each channel c used in p are also initialized, but the latter rather gets initialized to $levelOfChan(c)$. After initialization, instrumentation is given by the rules of Fig. 2. We now explain these rules.

$$\text{(S-Chan)}\ \frac{levelOfChan(nch) = \ell}{\Gamma \vdash nch : (int_\ell\ chan)_L} \qquad \text{(S-Int)}\ \frac{}{\Gamma \vdash n : int_L} \qquad \text{(S-Var)}\ \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \text{(S-read)}\ \frac{\Gamma \vdash c : int_\ell chan_{\ell_c}}{\Gamma \vdash \mathbf{read}\, c : int_{\ell \sqcup \ell_c}}$$

$$\text{(S-Op)}\ \frac{\Gamma \vdash e_1 : int_{\ell_1} \qquad \Gamma \vdash e_2 : int_{\ell_2}}{\Gamma \vdash e_1\ \mathbf{op}\ e_2 : int_{\ell_1 \sqcup \ell_2}} \qquad \text{(S-Skip)}\ \frac{}{\Gamma, pc, hc \vdash\ \mathbf{skip} :\ T, hc, \Gamma, \mathsf{skip}}$$

$$\text{(S-Assign)}\ \frac{\Gamma \vdash e : \sigma_{\ell_e}}{\Gamma, pc, hc \vdash x := e :\ T, hc, \Gamma[x \mapsto \sigma_{pc \sqcup \ell_e}],\ \mathsf{genassign}}$$

$$\text{(S-Send)}\ \frac{\Gamma(x) = int_{\ell_x} \qquad \Gamma(c) = (int_\ell\ chan)_{\ell_c} \qquad (pc \sqcup hc \sqcup \ell_x \sqcup \ell_c) \sqsubseteq_m \ell}{\Gamma, pc, hc \vdash \mathbf{send}\ x\ \mathbf{to}\ c : T, hc \sqcup \ell_c, \Gamma,\ \mathsf{gensend}}$$

$$\text{(S-If)}\ \frac{\begin{array}{c}\Gamma \vdash e : int_{\ell_e} \qquad h_3 = \sqcup_{j \in \{1,2\}} d(\Gamma, pc \sqcup \ell_e, cmd_j) \\ \bot \notin \mathrm{ran}(\Gamma_1 \sqcup \Gamma_2) \qquad h = (h_1 \sqcup h_2 \sqcup h_3 \sqcup level(t_1 \oplus_{\ell_e} t_2)) \\ \Gamma, pc \sqcup \ell_e, hc \vdash cmd_j : t_j, h_j, \Gamma_j, [\![cmd_j]\!] \quad j \in \{1,2\}\end{array}}{\Gamma, pc, hc \vdash\ \mathbf{if}\ e\ \mathbf{then}\ cmd_1\ \mathbf{else}\ cmd_2\ \mathbf{end} : (t_1 \oplus_{\ell_e} t_2), h, \Gamma_1 \sqcup \Gamma_2, \mathsf{genif}}$$

$$\text{(S-Loop)}\ \frac{\begin{array}{c}O(e, cmd, \Gamma \sqcup \Gamma') = t_o \qquad h = d(\Gamma, pc \sqcup \ell_e, cmd) \\ \ell_t = level(t) \qquad \ell_o = level(t_o) \qquad \bot \notin \mathrm{ran}(\Gamma \sqcup \Gamma') \qquad \Gamma \sqcup \Gamma' \vdash e : int_{\ell_e} \\ \Gamma \sqcup \Gamma', (pc \sqcup \ell_e), (hc \sqcup \ell_t \sqcup h') \vdash cmd : t, h', \Gamma', [\![cmd]\!]\end{array}}{\Gamma, pc, hc \vdash \mathbf{while}\ e\ \mathbf{do}\ cmd\ \mathbf{end} : t_o, h \sqcup h' \sqcup \ell_o, \Gamma \sqcup \Gamma', \mathsf{genwhile}}$$

$$\text{(S-Seq1)}\ \frac{\Gamma, pc, hc \vdash cmd_1 : D, h, \Gamma_1, [\![cmd_1]\!]}{\Gamma, pc, hc \vdash cmd_1; cmd_2 : D, h, \Gamma_1, [\![cmd_1]\!]}$$

$$\text{(S-Seq2)}\ \frac{t_1 \neq D \qquad \Gamma, pc, hc \vdash cmd_1 : t_1, h_1, \Gamma_1, [\![cmd_1]\!] \qquad \Gamma_1, pc, h_1 \vdash cmd_2 : t_2, h_2, \Gamma_2, [\![cmd_2]\!]}{\Gamma, pc, hc \vdash cmd_1; cmd_2 : t_1 \fatsemi t_2, h_2, \Gamma_2, [\![cmd_1]\!]; [\![cmd_2]\!]}$$

Fig. 2. Instrumentation and typing rules for the source language

Rules (S-Chan) and (S-Int) specify the channels type and integer constants. Rule (S-Var) encodes the typing of a variable, as given by environment Γ. Rule (S-Op) encodes expression typing and excludes channel operations. Rule (S-Read) specifies the current c value type. To prevent implicit flows, the specified security level takes into account the assignment context of channel variable c, hence the supremum $\ell \sqcup \ell_c$ Rule (S-Assign) specifies the type of x from the one of e to prevent explicit flows, and from pc, to prevent implicit flows. Its instrumentation is given by the following macro:

$$\mathsf{genassign} = \begin{cases} (x, x_{\mathsf{lev}}) := (e, \_\mathsf{pc} \sqcup e_{\mathsf{lev}}) & \text{if } \sigma = int \\ (x, x_{\mathsf{lev}}, x_{\mathsf{ch}}) := (e, \_\mathsf{pc} \sqcup e_{\mathsf{lev}}, e_{\mathsf{ch}}) & \text{if } \sigma = int_{\ell'}\, chan \end{cases}$$

The variable e_{lev} represents the level of expression e, as specified by Rule (S-Op). For example if $e = x + \mathbf{read}\ c$, then $e_{\mathsf{lev}} = x_{\mathsf{lev}} \sqcup c_{\mathsf{ch}} \sqcup c_{\mathsf{lev}}$. If $e = x + y$ then $e_{\mathsf{lev}} = x_{\mathsf{lev}} \sqcup y_{\mathsf{lev}}$. Rule (S-Send) requires $(pc \sqcup hc \sqcup \ell_x \sqcup \ell_c) \sqsubseteq_m \ell$. The four variables on the left-hand side correspond to the information level possibly revealed by the output to x_2. The instrumentation translates it as follows

$$\begin{aligned}\mathsf{gensend} = \ &\mathbf{if}\ \ \_\mathsf{pc} \sqcup \_\mathsf{hc} \sqcup x_{\mathsf{lev}} \sqcup c_{\mathsf{lev}} \sqsubseteq c_{\mathsf{ch}} \\ &\quad\ \mathbf{then}\ (\mathbf{send}\ x\ \mathbf{to}\ c)\ \mathbf{else}\ \mathbf{fail}\ \mathbf{end}; \\ &\_\mathsf{hc} := \_\mathsf{hc} \sqcup c_{\mathsf{lev}}\,;\end{aligned}$$

The halting context records the possible failure of the guarded send, it is updated with the assignment context of the channel. The following example illustrates why this is necessary.

```
if unknownValue > 0 (*H at runtime *)
   then c := lowChannel
   else c := highChannel  end;
send highValue to c;
send lowValue to lowChannel
```

Listing 1.4: Dangerous runtime halting

Assume that `unknownValue` is private and false at runtime. Then the first guarded send is accepted, but allowing an output on a low security channel subsequently would leak information about `unknownValue`. Updating $\_\mathsf{hc}$ will affect the check of all subsequent guarded send. Updating $\_\mathsf{hc}$ with x_{lev} or $\_\mathsf{pc}$ is not necessary since their value will be the same for all low-equivalent memories.

For the conditional rules, we need a union of environments that maps to each variable appearing in both branches the supremum of the two variable types, and for each channel variable appearing in both branches the security level U if the levels of their content differ.

Definition 3. *The supremum of two environments is given as* $dom(\Gamma_1 \sqcup \Gamma_2) = dom(\Gamma_1) \cup dom(\Gamma_2)$*, and*

$$(\Gamma_1 \sqcup \Gamma_2)(x) = \begin{cases} \Gamma_i(x) & \text{if } x \in dom(\Gamma_i) \backslash dom(\Gamma_j),\ \{i,j\} = \{1,2\} \vee \Gamma_1(x) = \Gamma_2(x) \\ (int_U\, chan)_{\ell_2 \sqcup \ell'_2} & \text{if } \Gamma_1(x) = (int_{\ell_1}\, chan)_{\ell_2} \\ & \quad \wedge \Gamma_2(x) = (int_{\ell'_1}\, chan)_{\ell'_2} \wedge \ell_1 \neq \ell'_1 \\ \sigma_{\ell \sqcup \ell'} & \text{if } \Gamma_1(x) = \sigma_\ell \wedge \Gamma_2(x) = \sigma_{\ell'} \\ \bot & \text{otherwise.} \end{cases}$$

The symbol $\bot$ is used to indicate that a typing inconsistency occured, e.g. when a variable is used as an integer in one branch and as a channel in another.

The function $level : \mathcal{T} \to \mathcal{L}$ returns the *termination level* (i.e., the level that termination depends on) and is defined as:

$$level(t) = \begin{cases} L \text{ if } t \in T, D \\ \ell \ \text{ if } t = M_\ell \end{cases}$$

Two operators are used to compose terminations types, $\oplus$, used in the typing of conditionals, and $\fatsemi$, used in the typing of sequences. They are defined as follows.

$$t_1 \oplus_\ell t_2 = \begin{cases} t_1 & \text{if } t_1 = t_2 \wedge [t_1 \neq M_L \vee \ell = L] \\ M_L & \text{if } \ell = L \wedge t_1 \neq t_2 \wedge \{t_1, t_2\} \subseteq \{T, D, M_L\} \\ M_H & \text{if } \ell = H \wedge [M_{\ell'} \in \{t_1, t_2\}, \ell' \in \mathcal{L} \text{ or } \{t_1, t_2\} = \{T, D\}] \\ M_{\langle \ell \sqcup \ell_1 \sqcup \ell_2 \rangle} & \text{otherwise, } t_1 = M_{\langle \ell_1 \rangle}, t_2 = M_{\langle \ell_2 \rangle} \end{cases}$$

where $\langle e \rangle$ is e, a level expression, without evaluation. We will evaluate $\langle e \rangle$ to U in the instrumentation type system (Fig. 2). We prefer to write $\langle e \rangle$ to emphasise the fact that U is the approximation of an expression.

$$t_1 \fatsemi t_2 = \begin{cases} M_{\ell_1 \sqcup \ell_2} & \text{if } t_1 = M_{\ell_1} \text{ and } t_2 = M_{\ell_2} \\ t_i & \text{if } t_j = T, \{i, j\} = \{1, 2\} \\ D & \text{otherwise} \end{cases}$$

The following example shows one more requirement. If in Listing 1.5 variable `unknownChannel` is a public channel at runtime, and if the last send command is reached and executed, it would leak information about `highValue`. The same leak would happen if instead of the guarded send we had a diverging loop.

```
if highValue > 0 then
   if ℓ ⊑ ℓ′
    then (send highValue to unknownChannel)
    else fail end;
end;
send lowValue to lowChannel
```

Listing 1.5: A guarded send can generate a progress leak

The following function $d : ((\mathcal{V} \uplus \mathcal{C} \rightarrow VarT) \times \mathcal{L} \times Cmd) \rightarrow \mathcal{L}$, is used to update the halting context, where Cmd is the set of commands. It approximates the information level that could be leaked through progress channels by a possibly failed guarded send in an unexecuted branch. Here, if $\Gamma(c) = (int_\ell\ chan)_{\ell'}$, then we write $\Gamma_{\mathsf{ch}}(c) = \ell$ and $\Gamma_{\mathsf{lev}}(c) = \ell'$.

$$d(\Gamma, pc, cmd) = \begin{cases} pc \sqcap (\bigsqcup_{c \in dc} (\overline{\Gamma_{\mathsf{ch}}(c)} \sqcup \Gamma_{\mathsf{lev}}(c))) & \text{if } dc \cap mv = \emptyset \\ pc & \text{otherwise} \end{cases}$$

In this definition, dc represents the set of *dangerous channels*, that is, the ones appearing in at least one guarded send in $[\![cmd]\!]$; mv is the set of variables that may be modified in cmd. Intuitively, if all the dangerous channels are of level H and not modified inside cmd, then we know that these guarded send cannot fail. If we cannot be sure of their level, then the halting context is updated with level pc. The supremum over the security levels of channels is taken in case the value of the channels is sensible (for example if `lowValue` was H in Listing 1.1).

(S-If) Its instrumentation is given by the following macro:

$$\begin{array}{l}\mathsf{genif} = \mathsf{\_oldpc}^{\nu}\ \texttt{:=}\ \mathsf{\_pc};\\ \quad \texttt{if}\ e\\ \quad\quad \texttt{then insIF(1,2)}\\ \quad\quad \texttt{else insIF(2,1)}\\ \quad \texttt{end};\\ \quad \mathsf{\_pc} := \mathsf{\_oldpc}\end{array} \quad \text{where } \texttt{insIF}(i,\texttt{other}) = \begin{array}{l}\mathsf{\_pc}\ \texttt{:=}\ \mathsf{\_pc} \sqcup e_{\mathsf{lev}};\\ \mathsf{hd}(\mathsf{\_hif}^{\nu}, mv_{\texttt{other}}, dc_{\texttt{other}});\\ [\![cmd_i]\!];\\ \mathsf{\_hc} := \mathsf{\_hc} \sqcup \mathsf{\_hif};\\ \mathsf{uphc}(t_1, t_2, e_{\mathsf{lev}});\\ \mathsf{update}(mv_{\texttt{other}})\end{array}$$

where dc_j represents the set of channels appearing in at least one **guardedSend** in $[\![cmd_j]\!]$, mv_j is the set of variables that may be modified in cmd_j, t_j is the termination type of cmd_j, e_{lev} is the guard condition's level expression and ℓ_e is the level of this guard (as computed by the typing system).

The instrumented code starts by saving the current context to $\mathsf{\_oldpc}^{\nu}$ (the symbol ν indicates that it is a fresh variable). The program context is updated with the security level of the guard condition. The **if** itself is then generated. In each branch, function hd, function d's at runtime, evaluates the information level possibly revealed by a failed guarded send in the other branch.

$$\mathsf{hd}(\mathsf{\_h}, mv, dc) = \begin{cases} \mathsf{\_h} := (\mathsf{\_pc} \sqcap (\bigsqcup_{c \in dc} \overline{c_{\mathsf{ch}}} \sqcup c_{\mathsf{lev}})) & \text{if } dc \cap mv = \emptyset \\ \mathsf{\_h} := \mathsf{\_pc} & \text{otherwise} \end{cases}$$

This must be computed before executing $[\![cmd_j]\!]$ because we want to evaluate whether the untaken branch could halt the execution or not. This must be done before $[\![cmd_j]\!]$ as the latter could modify the level of the dangerous channels.

Function uphc is used to generate the code updating the halting context.

$$\mathsf{uphc}(t_1, t_2, e_{\mathsf{lev}}) = \begin{cases} \mathsf{skip} & \text{if } t_1 = t_2 \in \{T, D\} \\ \mathsf{\_hc} := \mathsf{\_hc} \sqcup e_{\mathsf{lev}} & \text{otherwise} \end{cases}$$

The rational underlying uphc use is to protect the guard value from being revealed. If we know that both branches behave similarly, then the adversary will not be able to deduce private information. On the other hand, if the two branches may not behave the same way, then we have to perform $\mathsf{\_hc} := \mathsf{\_hc} \sqcup e_{\mathsf{lev}}$.

The following function updates the level variables of the untaken branch's modified variables so that they have similar types in all low-equivalent memories.

$$\mathsf{update}(mv) = \begin{cases} \mathsf{skip} & \text{if } mv = \emptyset \\ (x, x_{\mathsf{lev}}) := (x, x_{\mathsf{lev}} \sqcup \mathsf{\_pc}); \mathsf{update}(mv \backslash \{x\}) & \text{if } x \in mv. \end{cases}$$

In a situation like the following listing, this function permits to update `x`'s level, to protect `unknownValue`.

```
x := 0;
if unknownValue          (*H at runtime*)
  then x := 1  else skip end;
send x to lowChannel
```

Listing 1.6: Example illustrating why it is necessary to update the modified variables

(S-Loop) Typing the **while** involves a fixed point computation due to the flow sensitivity. It is easy to show that this computation converges. The typing relies on O, a statically called oracle computing the termination loop type (t_o).

$$
\begin{array}{l}
\mathsf{genwhile} = \mathsf{\_oldpc}^{\nu} \ \mathtt{:=}\ \mathsf{\_pc}\mathtt{;} \\
\qquad\qquad \mathtt{iWhile;} \\
\qquad\qquad \mathbf{while}\ e\ \mathbf{do} \\
\qquad\qquad\quad [\![cmd]\!] \\
\qquad\qquad\quad \mathtt{iWhile;}\ \mathbf{end}\mathtt{;} \\
\qquad\qquad \mathsf{\_pc} := \mathsf{\_oldpc}
\end{array}
\qquad \text{where} \quad
\begin{array}{l}
\mathtt{iWhile} = \\
\\
\\
\\
\\
\\
\end{array}
\begin{array}{l}
\mathsf{\_pc} := \mathsf{\_pc} \sqcup e_{\mathsf{lev}}\mathtt{;} \\
\mathsf{update}(mv)\mathtt{;} \\
\mathsf{hd}(\mathsf{\_hwhile}^{\nu}, mv, dc)\mathtt{;} \\
\mathsf{\_hc} := \mathsf{\_hc} \sqcup \mathsf{\_hwhile}\mathtt{;} \\
\mathsf{uphc}(t_o, t_o, e_{\mathsf{lev}})\mathtt{;}
\end{array}
$$

The inserted commands are similar to those of the **if**. The level variables and halting context are updated before the loop in case an execution does not enter the loop. They must be updated at the end of each iteration for the next iteration.

(S-Seq1) is applied if cmd_1 always diverges; we then ignore cmd_2, as it will never be executed. Otherwise, (S-Seq2) is applied. The halting context returned is h_2 instead of $h_1 \sqcup h_2$ because h_2 already takes into account h_1.

In a longer version, we present a type system for the target language. We show that a well typed program satisfies the progress-sensitive non-interference property 1, and that a program generated by our typing system is well typed.

5 Related Work

There has been much research in language-based techniques for controlling information flow over the last two decades.

Le Guernic et al. [8] present the first hybrid information-flow control monitor. The enforcement is based on a monitor that is able to perform static checks during the execution. The enforcement is not flow-sensitive but it takes into account concurrency. In Russo and Sabelfeld [11], the authors state that purely dynamic enforcements are more permissive than purely static enforcements but they cannot be used in case of flow-sensitivity. They propose a hybrid flow-sensitive enforcement based on calling static analysis during the execution. This enforcement is not progress sensitive.

Moore et al. [9] consider precise enforcement of flow-insensitive progress-sensitive security. Progress sensitivity is also based on an oracle's analysis, but they call upon it dynamically while we do it statically. We have also introduced additional termination types to increase the permissiveness of the monitor.

Chudnov and Naumann [5] inline a flow-sensitive progress-insensitive hybrid monitor and prove soundness by bisimulation. We inline a flow-sensitive progress-sensitive hybrid monitor, and we prove soundness using a mostly-standard security-type system for the target language.

Askarov and Sabelfeld [3] use hybrid monitors to enforce information security in dynamic languages based on on-the-fly static analysis. They provide a model to define non-interference that is suitable to progress-sensitivity and they quantify information leaks due to termination [2].

Askarov et al. [1] introduce a progress-sensitive hybrid monitoring framework where the focus is on concurrent programs, and the use of rely-guarantee reasoning to enable fine-grained sharing of variables between threads. Each thread is guarded by its own local monitor (progress- and flow-sensitive). Their local monitor could be replaced by a variant of our inlined monitor.

6 Conclusion

We have presented a hybrid information flow enforcement mechanism in which the main contributions are the following.

(a) Our monitor is one of the first hybrid monitor that is both flow- and progress-sensitive. It is more precise and introduces less overhead than currently available solutions (e.g., [9,10]). Since our monitor is inlined, it can be easily optimized using classical partial evaluation techniques, [7].

(b) We solve a few issues such as (1) the fact that it is not possible to approximate the level of a channel (by introducing a level U) and (2) the need to approximate the level of information that could be leaked through progress channels (by introducing a function d).

We believe our approach to be generalizable to complex lattices, but it will require a few alterations. Instead of only one uncertain level U, we would use sets of possible levels (U is, in some sense, an abstraction of the set $\{L, H\}$) that are ordered pointwise. That is, $\{L\} \sqsubseteq \{L, H\} \sqsubseteq \{H\}$. The function d would have to be adapted. Namely, the complement operation in d would have to be replaced with the following expression: $\{\ell : \ell \not\sqsubseteq \Gamma_{\mathsf{ch}}(c)\} \cap \{\ell : \ell \not\sqsupseteq pc\}$.

Future work includes extensions to concurrency, declassification and information leakage due to timing. We would like to scale up the approach to deal with real world languages and to test it on elaborate programs.

References

1. Askarov, A., Chong, S., Mantel, H.: Hybrid monitors for concurrent noninterference. In: Computer Security Foundations Symposium (2015)
2. Askarov, A., Hunt, S., Sabelfeld, A., Sands, D.: Termination-insensitive noninterference leaks more than just a bit. In: Proceedings of the European Symposium on Research in Computer Security: Computer Security (2008)
3. Askarov, A., Sabelfeld, A.: Tight enforcement of information-release policies for dynamic languages. In: CSF (2009)
4. Austin, T.H., Flanagan, C.: Efficient purely-dynamic information flow analysis. In: Proceedings of the Workshop on Programming Languages and Analysis for Security (2009)
5. Chudnov, A., Naumann, D.A.: Information flow monitor inlining. In: Proceedings of the 23rd IEEE Security Foundations Symposium (2010)
6. Cook, B., Podelski, A., Rybalchenko, A.: Proving program termination. Commun. ACM **54**(5), 88–98 (2011)
7. Jones, N.D., Gomard, C.K., Sestoft, P.: Partial evaluation and automatic program generation. Prentice Hall, Englewood Cliff (1993)
8. Le Guernic, G., Banerjee, A., Jensen, T., Schmidt, D.A.: Automata-based confidentiality monitoring. In: Okada, M., Satoh, I. (eds.) ASIAN 2006. LNCS, vol. 4435, pp. 75–89. Springer, Heidelberg (2008)
9. Moore, S., Askarov, A., Chong, S.: Precise enforcement of progress-sensitive security. In: CCS 2012 (2012)
10. O'Neill, K.R., Clarkson, M.R., Chong, S.: Information-flow security for interactive programs. In: CSFW. IEEE (2006)
11. Russo, A., Sabelfeld, A.: Dynamic vs. static flow-sensitive security analysis. In: CSF, pp. 186–199. IEEE Computer Society (2010)

12. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE J. Sel. Areas Commun. **21**(1), 5–19 (2003)
13. Smith, G., Volpano, D.: Secure information flow in a multi-threaded imperative language. In: POPL (1998)
14. Volpano, D., Irvine, C., Smith, G.: A sound type system for secure flow analysis. J. Comput. Secur. **4**(2), 167–187 (1996)
15. Zheng, L., Myers, A.C.: Dynamic security labels and noninterference. In: Dimitrakos, T., Martinelli, F. (eds.) Formal Aspects in Security and Trust. IFIP, vol. 173. Springer, Heidelberg (2005)

Privacy

Deducing User Presence from Inter-Message Intervals in Home Automation Systems

Frederik Möllers(✉) and Christoph Sorge

CISPA, Saarland University, Saarbrücken 66123, Germany
{frederik.moellers,christoph.sorge}@uni-saarland.de

Abstract. Privacy in Home Automation Systems is a topic of increasing importance, as the number of installed systems constantly grows. In this paper we investigate the ability of an outside observer to link sets of message timestamps together to predict user presence and absence. The question we try to answer is: *If attacker Eve has captured 1 hour of traffic from victim Alice's HAS and knows whether Alice was present at that time, can Eve deduce Alice's state by capturing another hour of traffic?* We apply different statistical tests and show that in certain situations, the attacker can infer the user's presence state with absolute confidence.

Keywords: Traffic analysis · Home automation · Statistical analysis · Privacy

1 Introduction

Home Automation Systems (HASs) are rapidly gaining popularity. They can be used to control lights, window blinds, heating etc. Wireless HASs are currently most popular for use in private homes, as the installation is easier than for their wired counterparts. However, previous research has shown privacy risks of wireless HASs [17]: A passive eavesdropper can derive information about user habits from message contents and metadata. Message encryption—the obvious countermeasure—does not prevent analysis of communication patterns: packets sent by a remote control to a door lock reveal that a user is locking or unlocking a door. Even if addresses are obfuscated, message intervals could reveal information, e.g. about absence or presence of users interacting with the HAS. In this paper, we study the extent of information leakage through message intervals in HASs. Our contribution consists of two parts: We present a new attack vector which passive adversaries can use to *infer information about the presence of users* from the timings of messages alone. Additionally, we analyse the success rates of this approach and determine conditions under which a high confidence can be achieved. While we acknowledge that certain situations are not distinguishable by software using our model (e.g. a user being asleep and not interacting with the system vs. a user being absent), the goal is to find out whether or not situations exist in which a correct statement can be reliably achieved.

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 369–383, 2016.
DOI: 10.1007/978-3-319-33630-5_25

The paper is structured as follows: In Sect. 2 we give an overview of existing research in similar areas. Section 3 contains the definition of our system model as well as our model of the attacker. In Sect. 4 we summarise the attack method whose effectivity we are investigating. Section 5 contains the description of our analysis procedure and is followed by its results in Sect. 6. We conclude the paper and provide an outlook on future work in Sect. 7.

2 Related Work

Several authors have pointed out the privacy risks of HASs. Jacobsson et al. provide an overview of security and privacy risks in HASs [8]. A survey by Denning et al. states "activity pattern privacy" with the sub-goals of "presence privacy" (which we investigate in the paper at hand) and protection of occupant identities as HAS security goals [5]. Privacy implications of specific systems have been studied by Mundt et al., who derived information about user habits from the communication of office building automation systems [14], and were able to eavesdrop on communication of a wired bus system from a distance of 5 cm [13]. In our own previous work, we have demonstrated the extent of information leakage from a wireless HAS that neither encrypts communication nor attempts to obfuscate sender and receiver addresses [17]. Moreover, we have studied legal aspects of HASs that use data processing in the cloud [16]. Packet inter-arrival times as a side channel have been considered by Wendzel et al. [19], but their work focuses on establishing covert channels. Our contribution instead addresses the problem of deducing information from existing timings.

As wireless HASs are a specific type of wireless sensor networks (WSNs), some general results about WSNs might apply. There is a considerable body of literature on privacy in WSNs. In their survey [10], Li et al. distinguish between data privacy (concerning both the queries and the sensed data) and context privacy, with the latter term referring to both *location privacy* and *temporal privacy*. A number of publications consider traffic analysis in WSNs as a means to breach location privacy [4,11,20], but this aspect is not very relevant in HASs. While temporal privacy (which concerns the ability of an attacker to determine the timing of an event detected by the WSN) is related to the problem we investigate, we do not consider individual events in the paper at hand.

The use of traffic analysis (i.e. analysis of traffic patterns without consideration of communication contents) is not restricted to particular networks; the distribution of message inter-arrival times is commonly considered in traffic analysis. For example, Moore and Zuev [12] use that distribution (among other discriminators) to classify internet traffic; Bissias et al. [2] use inter-arrival times to identify web sites in encrypted, proxied HTTP traffic. Celeda et al. [18] use traffic analysis in Building Automation Networks to detect attacks.

3 System and Attacker Model

For our analysis we assume the following situation. A user Alice has installed a home automation system. The system generates messages based on automation

rules and in reaction to user behaviour. Both the rules and Alice's habits are known only to Alice. As the idea of this paper is to analyse if certain information can be deduced from message timings alone, it makes sense to exclude all other possible sources of information an attacker might be able to use. Real-world observations as well as publicly known statistics (e.g. "The average user is asleep during the night and at work from 09:00 to 17:00.") are explicitly neglected here.

The network topology of our model is a fully connected graph with respect to intended communication. This means that any two devices which are intended to communicate with each other can do so directly. This model is used in many available products; only few systems employ multi-hop communication. However, the research presented in this paper can be used as a base for developing dummy traffic schemes in both types of systems.

The communication is fully encrypted and packets are padded to a fixed length. Both message payloads and message headers (including source and destination addresses) are hidden from an outside observer.

In certain situations, low-level channel information can be used to try and fingerprint devices when both sender and receiver are static [1,3,6]. For the analysis at hand, we disregard this possible source of information. We argue that these kinds of attacks require a level of effort and dedication from the attacker which is unrealistic for common houses or when mounting traffic analysis attacks on a large scale against many buildings at once. Furthermore, countermeasures against these attacks have been explored in literature. We thus assume that the attacker cannot determine the source of a packet by these means.

We model our attacker—Eve—as a global passive adversary. Eve can detect any communication happening within the network, i.e. she can capture any packet being transmitted. However, Eve cannot break the packet encryption and she cannot distinguish between different devices by other means such as triangulation or wireless device fingerprinting.

Eve's goal is to determine whether or not Alice is at home at a given time. For this, we assume she has the following *a priori* information about Alice's home automation system:

1. Eve knows that Alice's HAS does not generate dummy traffic.
2. Eve has captured all communication packets during one hour of HAS operation. She also knows whether Alice was at home during this time.

The reason why we choose an interval of one hour for item 2 is twofold. On the one hand, a time frame of more than one hour allows Eve to mount sophisticated device fingerprinting attacks [6], invalidating our assumptions. However, it also makes decisions less useful: The longer the time frame, the less likely Alice is to keep this state during the next minutes or hours. On the other hand, a shorter time frame makes decisions harder, as there is less data to base an assumption on. We performed the same experiments with time frames of half an hour and two hours, getting nearly the same results: The difference in the AUC values in Sect. 6.3 was 0.005 on average with a maximum of 0.104.

Using the available information, Eve needs to decide at a given time whether Alice is currently at home or not. Eve can capture the communication packets again for the same time frame to try and deduce Alice's presence state.

4 Attack Methodology

Our analysis works as follows: We assume the role of the attacker, Eve. Using the captured communication packets from two different time frames of one hour each, we try to find similarities in the statistical distribution of timestamps or inter-message intervals. We apply three different statistical tests to the two samples: The Kolmogorow-Smirnow Two-Sample Test [9], the Chi-Square Test of Independence [15] and the "Message Counts Test".

The statistical tests used here tackle the null hypothesis that *the two samples have the same underlying distribution function*. Instead of rejecting the null hypothesis with a certain confidence at a threshold depending on the desired confidence, we analyse the computed test statistics and try to determine suitable thresholds ourselves. The reason behind this is twofold: On the one hand, we do not have any a priori knowledge about the underlying distribution functions. On the other hand, we want to determine whether the difference in the distributions between two samples with different user states is high enough to allow a distinction based on the calculated test statistics. If this is the case, we can subsequently calculate thresholds and resulting confidence values for HASs.

4.1 Kolmogorow-Smirnow Test (KS Test)

The Kolmogorow-Smirnow Test for homogeneity [9] is based on the empirical cumulative distribution functions of the two input samples. Informally speaking, it measures the maximum vertical distance between the two curves. Formally, given to samples $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_m]$ with respective empirical cumulative distribution functions F_X and F_Y, it computes the value

$$D = \sup_a |F_X(a) - F_Y(a)| \tag{1}$$

If the result D is high, the null hypothesis is rejected.

We use the `SciPy`[1] implementation of the KS 2-sample test from `SciPy` version 0.14.0 and apply it to the inter-message time intervals. In addition to the KS statistic D (sometimes referred to as $d_{\max}$ or $D_{a,b}$ in literature), the implementation computes a *p-value* as a function of D and the sample sizes. This accounts for the fact that large samples with the same underlying distribution are expected to show less differences than smaller samples (as per the law of large numbers). We examine both the value of D as well as the *p-value*.

[1] http://www.scipy.org, accessed 2015-12-18.

4.2 Chi-Square Test (χ^2 Test)

Pearson's Chi-Square Test [15] follows a similar approach as the KS test, but calculates the sum of squared differences between the actually measured frequencies and the expected ones. In the 2-sample form, the expected frequencies are estimated by taking the average frequencies of the two samples. Formally, the test expects categories and respective frequencies as inputs. Given two samples and m categories, these frequencies can be written as $X = [x_1, x_2, \dots x_m]$ and $Y = [y_1, y_2, \dots, y_m]$, where x_i is the number of elements in the first sample which fall into the i-th category. Using the intermediate definitions

$$n = n_x + n_y = \sum_{i=1}^{m} x_i + \sum_{i=1}^{m} y_i \tag{2}$$

$$\forall z \in \{x, y\} : E_{z,i} = \frac{n_z \times (x_i + y_i)}{n} \tag{3}$$

the test statistic is then defined as

$$\chi^2 = \sum_{i=1}^{n} \frac{(x_i - E_{x,i})^2}{(E_{x,i})} + \sum_{i=1}^{n} \frac{(y_i - E_{y,i})^2}{E_{y,i}} \tag{4}$$

If the value of χ^2 is high, the null hypothesis ("The two samples have the same underlying distribution function.") is rejected.

For the Chi-Square Test, we use a custom implementation. Similar to the Kolmogorow-Smirnow Test, it is applied to the inter-message time intervals.

As the test expects the two samples to be categorized into bins, we need to do this before calculating the actual test statistic. Literature suggests to choose bin sizes so that no bin contains less than 5 elements for any sample [7]. Thus, we adaptively choose bins of varying size. The lower bound for the first bin is the lowest value in any of the two input samples. The upper bound for a bin (which is also the lower bound for the next bin) is chosen as the smallest number which results in at least 5 elements of each sample falling into this bin. We thus guarantee that at least 5 values are in each bin for each sample. An example for the binning approach is depicted in Fig. 1. For the Chi-Square Test we calculate and examine the test statistic.

Sample A: 1 1 1 2 2 | 3 3 3 4 4 4 4 5 | 6 6 6 6 7 7 8 8
Sample B: 1 1 1 1 2 2 2 | 3 3 3 4 5 | 6 6 6 6 6 7 7 7

Fig. 1. Example of the approach used for binning using a minimum bin size of 5. The bounds are chosen so that at least 5 elements of each sample fall into one bin.

4.3 Message Counts Test (MC Test)

Our "Message Counts Test" divides the number of messages in the larger sample by the number of messages in the smaller one and subtracts 1, resulting in a value within $[0, +\infty[$. Higher values indicate larger differences in the amounts of messages, just as higher results in the other tests indicate different distributions. The idea behind it is that if the sheer amount of activity in the system is very different to that during the reference time frame, the user state is likely to be different. For example, if the reference capture was taken while Alice is present and the capture in question shows lower activity, Alice is likely to be absent.

Formally, given two samples $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_m]$, the test statistic is defined as

Similar to the Chi-Square test, we calculate and examine the test statistic.

$$C = \frac{\max(n, m)}{\min(n, m)} - 1 \tag{5}$$

5 Analysis Procedure

We obtained input data for our analysis by collecting packet captures from two real-world home automation systems. *System 1* is an installation using *Home-Matic* hardware, an off-the-shelf solution for consumers, which was already used as a base for our previous work on this topic [17]. The owner voluntarily captured all traffic for 36 days and provided us with the log files as well as presence and absence times. *System 2* is data from a custom system, built by combining multiple automation products from different manufacturers. Traffic was recorded for 37 days and published in a series of news articles[2].

As a first step, we annotate each message with the user state: PRESENT and ABSENT are chosen based on the available data. A third state, ASLEEP is introduced to handle the fact that during night hours (22:00 to 08:00), users are usually asleep and thus the activity of the system is reduced. Due to the vague nature of the ASLEEP state and the fact that we cannot be sure whether the users were actually asleep during this time, we exclude it from further analysis and only investigate messages whose state is either PRESENT or ABSENT.

Analysing each system by itself, we construct (non-overlapping) intervals of 1 hour each and during which the user state did not change. For each interval, we gather the messages sent during this time into *Message Groups*. Each Message Group is thus identifiable by its system and the timestamp of the first message. Also, as per the construction of intervals described above, each group has a fixed user state. For System 1, we obtain 180 Message Groups with state PRESENT, 136 Message Groups with state ABSENT and 237 Message Groups with state ASLEEP. For System 2, the numbers are 223, 125 and 296, respectively.

For all non-identical combinations of Message Groups (only considering those with states PRESENT and ABSENT)—167,941 in total—we perform the 3 statistical tests mentioned in Sect. 4. We then visualize the results in boxplots, both

[2] http://spon.de/aeDkn, accessed 2015-12-18.

overall per system as well as individually for each combination of user states. In a second step, we test different thresholds for all tests and plot the true and false positive rates in ROC diagrams.

6 Analysis Results

6.1 Test Suitability in the General Case

At first, we plot all test results by system and test and only distinguish between the two cases whether or not the samples have different user states. This section gives a general and quick overview over the suitability of the tests for our purposes. If the plots of the two cases differ significantly, the test results carry a high amount of information and if they are largely the same, the information immediately available from the test result is limited. The plots are visualised in Fig. 2 for both systems. The boxplots do not show any immediately obvious peculiarities. For both systems and all tests, the boxes overlap and thus suggest that the tests cannot be used as a universal oracle telling Eve whether the 2 compared samples have been taken with the same user state.

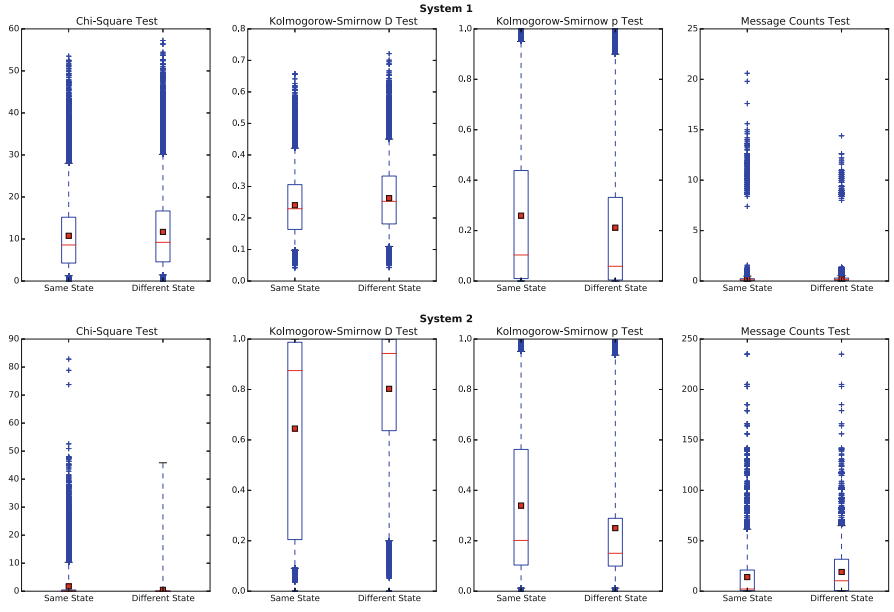


Fig. 2. General test results for both systems. The boxes extend from the first to the third quartile. The whiskers extend up to $1.5 \times IQR$ past the boxes, where IQR is the interquartile range. If $IQR = 0$ (as with the χ^2 Test for different states in System 2), the whiskers extend up to the minimum and maximum values. Red lines mark the medians while red squares mark the arithmetic means. Blue plus signs show outliers beyond the whiskers (Color figure online).

System 1. For System 1, the χ^2 Test values are broadly spread. Comparing samples with the same user state yields values from 0 to 53.5, samples from different states lead to values from 0 to 57.2. This suggests that there may be an upper bound to the value for samples of the same state and that values above this limit indicate a different state of the two compared samples.

The *KS Test* statistic D ranges from 0.04 to 0.66 for the same state and from 0.04 to 0.72 for different states. Like the Chi-Square test, this suggests an upper bound for the value in the same-state case.

The *KS p-values* again provide similar information. For the same state, the values range from 6.41×10^{-12} to $1 - 10^{-11}$, for different states they range from 4.76×10^{-15} to $1 - 10^{-12}$. The null hypothesis ("The two samples originate from the same distribution [=the same state].") is rejected for p-values lower than a threshold. The lower minimum value for different results shows that the default thresholds are not useful in our scenario.

The *MC Test* provides the least useful results. The values are in fact misleading: While they range from 0 to 20.6 for samples with the same state and the minimum is the same for different states, the maximum value in the latter case is only 14.4. This shows that while the user state does not change, the number of messages being generated in a given time frame can differ significantly.

System 2. The results for System 2 offer much less information than those for System 1. The χ^2 Test values range from 0 to 82.8 for samples with the same state and from 0 to 45.8 for samples with different states. As shown in Fig. 2, 75 % (the lower three quartiles) of the tests with different states had the result 0. These values are misleading if interpreted in the same way as those of System 1. Intuitively, the values should be higher for different states (and they are for System 1). We conclude that either the test's usefulness depends on the type of the HAS or that the previous results were not representative.

The *KS Test statistic D* yields values in the full range $[0, 1]$ for samples with the same state. While this already indicates that the test is not useful for this system, the same range of values for samples with different states support this.

Consequently, the *KS Test p-values* are inconclusive as well: They range from 1.13×10^{-16} to 1 for the same state and from 2.28×10^{-8} to 1 for different states.

The *MC Test* surprisingly yields the exact same range of values for both cases: The results range from 0 to 235 in both cases.

6.2 Test Suitability per State Pair

In the next step we take a closer look at the different combinations of user states. Our hypothesis is that the tests may give useful results for certain combinations of states and less useful results for others. This section deals with the performance of the tests for a given pair of user states. Figure 3 summarizes the results for both systems.

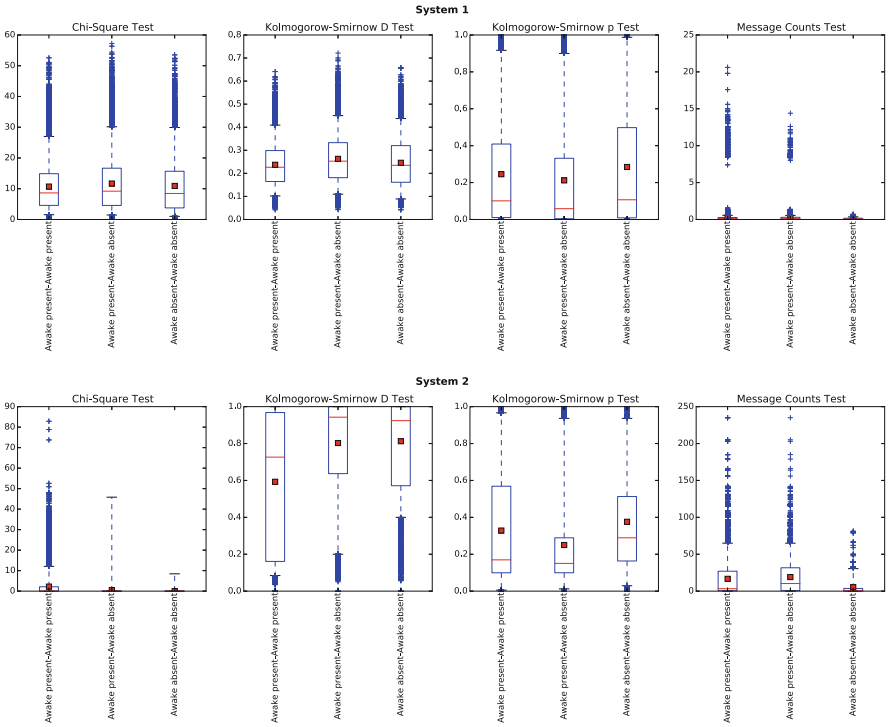


Fig. 3. Per state pair test results for both systems. The plot parameters are the same as for Fig. 2. The reason why the combination PRESENT-ABSENT does not appear is the symmetry of all tests: $T(a, b) = T(b, a)$.

System 1. Results of the χ^2 Test for System 1 do not yield much more information than what we could already see from the general evaluation. The two cases in which the user states are the same largely overlap and the ranges of values are almost the same. The same holds for both statistics of the *KS Test*. The *MC Test* provides some new results: If both samples have the state ABSENT, the values do not go above 0.71. This means that if Eve obtains a sample known to have the state ABSENT, and gets higher value when comparing it to a second (unknown) sample, she can be sure that Alice was present during the time frame of the second sample. However, this is only the case for 2.4 % of the tested ABSENT-PRESENT sample pairs.

System 2. In contrast to System 1, the boxplot of the χ^2 Test for System 2 exhibits obvious differences between the state pairs. If one of the samples has the state ABSENT, 75 % of the tests evaluate to 0. Similarly to the MC Test for System 1, the plots show that there is a threshold above which Eve can be sure that Alice is PRESENT if her first (known) sample has the state ABSENT. This threshold is at 8.45 and 2.12 % of the ABSENT-PRESENT pairs reach a higher

value. However, also similar to System 1, Eve cannot make such a confident decision if her known sample has state PRESENT.

The *KS Test* does not show such features; this is consistent with System 1.

The *MC Test* confirms the observation from the χ^2 Test and yields another threshold. The threshold value is 81 and 1.77 % (494 out of 27,875) of the tests with different states result in higher values. Surprisingly, though, none of these 494 Message Group pairs gave a result above the threshold for the χ^2 Test. In fact, some pairs even evaluated to 0 in the χ^2 Test. This is highly interesting, as it suggests that a combination of different tests with the same input data can provide significantly more information than one test alone. Using the thresholds of both tests, Eve can identify 3.89 % or 1084 of 27,875 Message Group pairs as having different states if one of the samples is known to have the state ABSENT.

6.3 The Effect of Different Thresholds on Classification Rates

As shown in the previous section, some tests exhibit maximum values for certain state combinations, and knowing such values may enable Eve to infer Alice's user state at a given time with absolute confidence. Below these, however, statements about presence and absence are more difficult to make. In this section we examine the effect of different chosen threshold values on the classification rates.

We compute True and False Positive Rates TPR and FPR for all possible threshold levels using the data from the tests previously conducted. In our case, the rates are defined as follows:

If $s(a)$ is the state of a sample a, $T(a, b)$ is the test result of the pair (a, b), t is the threshold value below which sample pairs are classified as having the same state and $N_{a,b}(cond)$ is the number of sample pairs a, b which satisfy a condition $cond$, then

$$TPR = \frac{N_{a,b}(s(a) = s(b) \wedge T(a, b) < t)}{N_{a,b}(s(a) = s(b))} \tag{6}$$

$$FPR = \frac{N_{a,b}(s(a) \neq s(b) \wedge T(a, b) < t)}{N_{a,b}(s(a) \neq s(b))} \tag{7}$$

TPR is the number of correctly classified same-state pairs divided by the total number of same-state pairs and FPR is the number of different-state-pairs which were incorrectly classified as having the same state divided by the total number of different-state pairs. TPR is a measure for how well the test can identify samples with the same state as the source and FPR is a measure for how often the test falsely reports two samples for having the same state.

In order to visualise the rates, we plot ROC (Receiver Operating Characteristics) curves and calculate the AUC (Area Under Curve) for all of them. ROC curves illustrate how fast the test performance drops (i.e. how fast the False Positive Rate increases) when raising the threshold to get a higher True Positive Rate. The AUC is a numerical measure for this quality: In the ideal case (the test has a TPR of 1.0 and a FPR of 0.0) the value is 1 and in the worst case (the test does not perform better than randomly guessing), the value is 0.5. Values

below 0.5 are similar to values above, since the test result interpretation can be inverted to invert the ROC curve (i.e. values *above* the threshold are interpreted as indicators for a same-state pair).

A selection of ROC curves is depicted in Fig. 4. Some tests (most notably the χ^2 Test for System 2) yield high values for both rates with the lowest possible threshold, which is why the curves do not start at the origin $[0, 0]$. To calculate the AUC for these cases, we use the *line of no-discrimination*—the values obtained by randomly guessing—up to the FPR of the lowest threshold (the X coordinate). From there on, we proceed with the regular estimation and calculate the area below the straight line between two subsequent data points.

Most curves do not exhibit large deviations from the mean line. For System 1, both the χ^2 Test and the two KS Tests yield an AUC between 0.52 and 0.57. Only the MC Test performs slightly better, the AUC is 0.525 for a source sample with state PRESENT and 0.688 for an ABSENT source sample (shown in Fig. 4).

Overall, the results for System 1 suggest that statistical tests are only of limited use in deducing user states from inter-message intervals.

System 2 mostly confirms this observation, although the performance of the different tests varies drastically.

The χ^2 Test performs badly: For a PRESENT source sample, the minimum obtainable False Positive rate is 91.6 % at a True Positive Rate of 61.3 % (the threshold value in this case is 0). For an ABSENT source sample, the minimum False Positive rate is consequently the same, but the minimum True Positive rate is 98.0 %. The KS Test and the MC Test perform much better. Their AUC values are relatively high and significant True Positive rates can be obtained while keeping the False Positive rates below 50 %.

From the analysis of the ROC curves we draw two conclusions. Firstly some tests exhibit a significant deviation from the line of no-discrimination. Combining multiple tests could further improve the results and yield more information. Secondly we can confirm our previous observation that extreme threshold values lead to absolute certainty in the classification.

6.4 Feasibility of Detection in Practice

The statistical tests do not yield clear results in all cases we examined. However, upper or lower bounds can be determined in some cases, which then allow Eve to make statements with absolute confidence. The requirements for these thresholds to be useful for Eve are not hard to meet: She needs a source sample which—when tested in conjunction with samples of a different state—yields values above or below the thresholds.

To verify the practicability of this attack we divide our traffic data into a training set and a test set. For training, we use the first 70 % of our data (221 Message Groups from System 1, 244 Message Groups from System 2).

We perform all aforementioned tests on the training data and calculate thresholds for Message Group pairs with the same state. Using these threshold, we choose one Message Group for every system and state where the amount of correct classifications among the training data is maximized—i.e. the Group

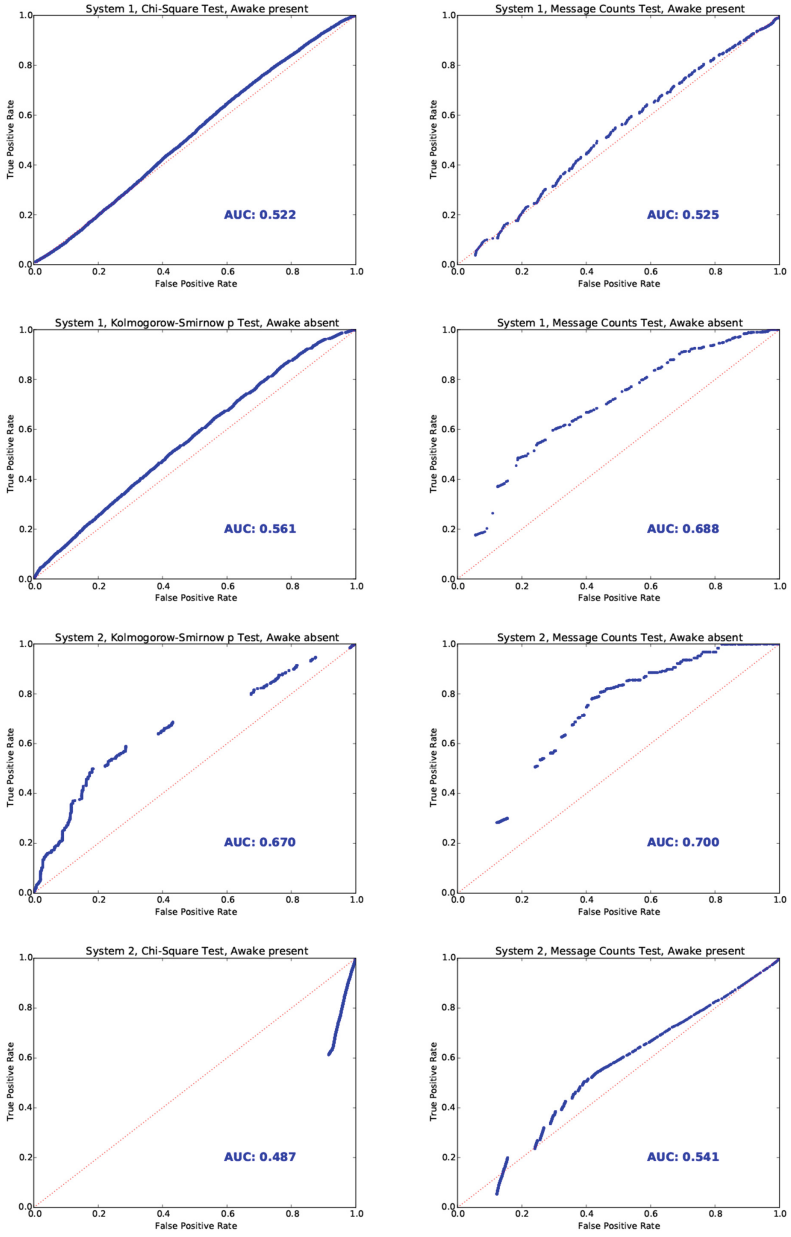


Fig. 4. ROC curves for different tests and source states. Blue points show the actual values, dotted red *lines of no-discrimination* show linear ascension from $[0, 0]$ to $[1, 1]$—the values obtained by randomly guessing. The graphics indicate that the test performance strongly depends on the system and the source sample. As noted in Sect. 6.1, the χ^2 Test for System 2 produces counterintuitive results (Color figure online).

with the highest TPR among the training data. We then check each of these Groups against the test data and calculate True and False Positive Rates using the thresholds calculated from the training data before.

For System 1 using an ABSENT source sample, we reach a TPR of 5.3 % and a FPR of 1.1 %. This suggests that the attack is not useful in practice. Using a PRESENT source sample, however, the FPR is at 0 while the TPR reaches 1 %. It is thus only a matter of time until Eve can successfully identify an ABSENT sample if she has a suitable PRESENT source sample. For System 2, the best ABSENT source sample achieves a TPR of 5.8 % while the FPR also stays at 0. However, in the data for this System no suitable PRESENT source sample exists. The tests do not yield thresholds which allow for an unanimous classification.

This particular attack is not likely to be encountered in reality: Eve would have to manually observe Alice's home for several hours or even days, annotating the captured traffic with the user states for every one-hour sample. However, the experiment shows that under the right circumstances, unanimous classification is possible. The experiment supports the theory that system-wide thresholds exist which allow for a classification of states with absolute certainty. The follow-up question whether such thresholds exist for a manufacturer or production series remains to be answered.

7 Conclusion and Outlook

In this paper we have performed the first analysis of inter-message intervals in Home Automation using statistical goodness of fit tests. We have used sample data from two real world installations to measure the ability of an attacker in deducing user states. In particular, we tried to answer the question:

If Eve has captured 1 hour of traffic from the Alice's HAS and knows whether Alice was present at that time, can Eve deduce Alice's state by capturing another hour of traffic?

Comparing and combining various tests, we were able to identify conditions under which the question above could be confidently answered with *yes*.

The χ^2 Test provides little information with regard to the question. However, the MC Test and, in some cases, the KS Test reveal identifiable discrepancies between samples with different states. A combination of all three tests allow an attacker to mount a practical attack on the system and infer the user state by passively listening after obtaining a suitable source sample.

For future work, we will work on new tests and combine them with those applied in this paper in order to obtain more information. At the same time, we will study the different properties of HASs to find out if there are filtering techniques which can be applied to the samples in order to make the tests more effective. Since this increases the abilities of an attacker to predict user presence and absence without physical labour, we will also develop dummy traffic schemes for use in HASs. These offer users the ability to mask their traffic and hide their state from unauthorized observers.

References

1. Bagci, I.E., Roedig, U., Schulz, M., Hollick, M.: Gathering tamper evidence in Wi-Fi networks based on channel state information. In: Proceedings of ACM WiSec 2014, pp. 183–188. ACM, New York (2014)
2. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy vulnerabilities in encrypted HTTP streams. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 1–11. Springer, Heidelberg (2006)
3. Brik, V., Banerjee, S., Gruteser, M., Oh, S.: Wireless device identification with radiometric signatures. In: Proceedings ACM MobiCom 2008, pp. 116–127. ACM, New York (2008)
4. Deng, J., Han, R., Mishra, S.: Counter measures against traffic analysis attacks in wireless sensor networks. In: Proceedings IEEE/CreateNet SecureComm 2005, pp. 113–126 (2005)
5. Denning, T., Kohno, T., Levy, H.M.: Computer security and the modern home. CACM **56**(1), 94–103 (2013)
6. Desmond, L.C.C., Yuan, C.C., Pheng, T.C., Lee, R.S.: Identifying unique devices through wireless fingerprinting. In: Proceedings ACM WiSec 2008. pp. 46–55. ACM, New York (2008)
7. Fisher, R.A., Yates, F.: Statistical Tables for Biological, Agricultural and Medical Research, 6th edn. Oliver and Boyd, Edinburgh (1963)
8. Jacobsson, A., Boldt, M., Carlsson, B.: A risk analysis of a smart home automation system. Future Generation Computer Systems **56**, 719–733 (2016)
9. Kolmogorow, A.N.: Sulla determinazione empirica di una legge di distributione. Giornale dell'Istituto Italiano degli Attuari **4**, 1–11 (1933)
10. Li, N., Zhang, N., Das, S.K., Thuraisingham, B.: Privacy preservation in wireless sensor networks: A state-of-the-art survey. Ad Hoc Netw. **7**(8), 1501–1514 (2009)
11. Li, Y., Ren, J.: Source-location privacy through dynamic routing in wireless sensor networks. In: Proceedings IEEE INFOCOM, pp. 1–9 (2010)
12. Moore, A.W., Zuev, D.: Internet traffic classification using bayesian analysis techniques. In: Proceedings ACM SIGMETRICS 2005, pp. 50–60. ACM, New York (2005)
13. Mundt, T., Dähn, A., Glock, H.W.: Forensic analysis of home automation systems. In: HotPETs 2014 (2014)
14. Mundt, T., Kruger, F., Wollenberg, T.: Who refuses to wash hands? Privacy issues in modern house installation networks. In: IEEE BWCCA 2012, pp. 271–277 (2012)
15. Pearson, K.: On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. In: Kotz, S., Johnson, N. (eds.) Breakthroughs in Statistics, pp. 11–28. Springer Series in Statistics, Springer New York (1992)
16. Möllers, F., Sorge, C.: Hausautomationssysteme im Datenschutzrecht. In: Cooperation: Proceedings of the 18th Legal Informatics Symposium IRIS 2015, pp. 553–558. Österreichische Computer Gesellschaft, German (2015)
17. Möllers, F., Seitz, S., Hellmann, A., Sorge, C.: Extrapolation and prediction of user behaviour from wireless home automation communication. In: Proceedings of ACM WiSec 2014, pp. 195–200. ACM, New York (2014)
18. Čeleda, P., Krejčí, R., Krmíček, V.: Flow-based security issue detection in building automation and control networks. In: Szabó, R., Vidács, A. (eds.) EUNICE 2012. LNCS, vol. 7479, pp. 64–75. Springer, Heidelberg (2012)

19. Wendzel, S., Kahler, B., Rist, T.: Covert channels and their prevention in building automation protocols: a prototype exemplified using bacnet. Proc. IEEE GreenCom **2012**, 731–736 (2012)
20. Yao, L., Kang, L., Shang, P., Wu, G.: Protecting the sink location privacy in wireless sensor networks. Personal and Ubiquitous Comput. **17**(5), 883–893 (2013)

Privacy by Design Principles in Design of New Generation Cognitive Assistive Technologies

Ella Kolkowska[1(✉)] and Annica Kristofferson[2]

[1] School of Business, Örebro University, Örebro, Sweden
ella.kolkowska@oru.se
[2] School of Science and Technology, Örebro University, Örebro, Sweden
annica.kristoffersson@oru.se

Abstract. Today, simple analogue assistive technologies are transformed into complex and sophisticated sensor networks. This raises many new privacy issues that need to be considered. In this paper, we investigate how this new generation of assistive technology incorporates Privacy by Design (PbD) principles. The research is conducted as a case study where we use PbD principles as an analytical lens to investigate the design of the new generation of digitalized assistive technology as well as the users' privacy preferences that arise in use of this technology in real homes. Based on the findings from the case study, we present guidelines for building in privacy in new generations of assistive technologies and in this way protect the privacy of the people using these technologies.

Keywords: Privacy requirements · Privacy by design · Assistive technology · Cognitive decline · Aging in place

1 Introduction

Assistive technology (AT) is a broad term used to describe any item, object, device or system that enables disabled people to perform a task that they would otherwise be unable to do, or increase the ease and safety by which certain tasks can be performed. AT plays an important role in supporting elderly people in living independently at home [1]. In this paper, we focus on AT suitable for elderly with a mild cognitive decline, e.g., dementia. The rapid development of cognitive assistive technologies (CAT) paves the way for new and more efficient solutions that improve the quality of life for people being affected by cognitive decline while decreasing their caregivers' burden of care [2]. Today, we are rapidly moving from analogue CAT accessible only for the user of the specific device to digital replicas, and further extensions of the CAT involving sensor networks but also technologies accessible by remote caretakers. In this paper, we refer to this technology as *the new generation of CAT*. The development of CAT raises many new privacy issues which need to be considered [3–5] but unfortunately, most of today's development projects are technically oriented and focus on functionality and technical effectiveness of the developed solutions [5, 6]. Consequently, the privacy of the user is often not sufficiently considered during development and implementation of the new generation of CAT [5].

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 384–397, 2016.
DOI: 10.1007/978-3-319-33630-5_26

The concept of *Privacy-by-Design (PbD)* advocated by EU [7] aims to ensure privacy protection and personal control over the information collected when IT systems are used. PbD principles are formulated to support the designers in taking the protection of privacy into account during the development of technologies (such as CAT) that in turn results in better privacy protection for the users of these technologies.

In this paper, we investigate how the *new generation of CAT* incorporates PbD principles with respect of elderly users' privacy requirements. Based on our findings, we formulate a set of guidelines for building in elderly users' privacy preferences into CAT and in this way protect the privacy of the people using them. A starting point for this research is a set of PbD principles suggested by Cavoukian et al. [8] for the context of personal health monitoring. The main contribution of the paper is adding a users' perspective to the existing technology-focused PbD principles.

The paper is structured as follows. Section 2 describes and discusses the PbD concept and PbD principles. Section 3 describes the new generation of CAT at focus in this case study. Section 4 presents our research method. Section 5 reports on our analysis of the case study. In Sect. 6, we discuss the results and present a set of guidelines for building into privacy in the new generation of CAT. Section 7 concludes the paper.

2 Privacy by Design Principles

Protection of privacy regarding sensitive personal data, is legally regulated in most countries and therefore cannot be overlooked in design and use of the new generation of CAT. It is also recognized that people, generally are not capable or not interested in protecting their own sensitive information, thus there is a need for standardization and automatization of privacy protection [7]. PbD is a way of embedding privacy into the design specifications of technologies. Cavoukian et al. [8] suggest seven PbD principles for the context of personal health monitoring These seven principles are an adjustment of the general OECD "Guidelines on the Protection of Privacy and Transborder Flows of Personal Data" [8]. The seven PbD principles formulated by Cavoukian et al. for the context of health care monitoring are [8]:

1. Proactive not Reactive; Preventative not Remedial. The PbD approach is characterized by proactive rather than reactive measures. The first principle emphasizes that it anticipates and prevents privacy invasive events before they happen.

2. Privacy as the Default Setting. The second principle means that no action is required by the users to protect their privacy because it is built into the system, by default.

3. Privacy Embedded into Design. The third principle emphasizes the importance of embedding privacy into the design and architecture of IT systems and business practices from the beginning and not add it afterwards.

4. Full Functionality - Positive-Sum, not Zero-Sum. The fourth principle considers privacy as an integral part of the system without diminishing its functionality.

5. End-to-End Security - Full Lifecycle Protection. The fifth principle relates to the life cycle management of information and stresses that data should be protected in all data handling from its beginning (collection) to its end (destruction). I.e., this principle is important to ensure privacy of the people using the technology when it is in use.

6. Visibility and Transparency - Keep it Open. The sixth principle states that data protection should be visible and transparent to the different stakeholders, e.g. users and providers. In the context of our case study, this principle means that the users are informed about what data is being collected and for what purpose, how the data is being used, and who can access it.

7. Respect for User Privacy - Keep it User-Centric. This principle means that the individual's privacy should be an interest of designers and operators of health systems.

Cavoukian et al. [8] define information privacy as an individual's ability to exercise control over the collection, use, disclosure and retention of his or her personal information, including personal health information. Nordgren [9] argues that the PbD principles suggested by Cavoukian et al. are supportive in ensuring privacy of the patients in the context of personal health monitoring, although they have two limitations: (1) PbD cannot solve all privacy problems because responsible handling of information by human agents is also important, and (2) automated data protection is useful in many cases, but it is not desirable in all cases. Generally, socially-oriented research on privacy in this context is sparse. Previous research shows that elderly people's privacy preferences are not sufficiently investigated in development of CAT [5, 10, 11]. Usually, it is assumed-at least implicitly-that a "common" CAT user does not care about privacy [11]. Consequently, development of such technologies is often functionality-driven without taking care of privacy issues that arise in use of such technologies in real settings [5, 6]. That is against the PbD thinking. Our interest lies in studying the privacy concerns that arise in use of CAT in real settings as well as their potential, in case of unforeseen usage in the future. Since the importance of respecting the users' privacy is especially empathized in the seventh PbD principle we decided to conduct a case study focusing on the users' privacy preferences that come up in use of CAT in relation to Cavoukian et al.'s seven PbD principles.

3 System Description

An example of the new generation of CAT is HOMEbasic[1], which is a "safety and security" package for those who need a combination of time and memory support, environmental monitoring and alarm functionalities. HOMEbasic in its standard package consists of a MEMOplanner and a sensor network (a door magnet, a stove sensor, an on/off sensor, two motion sensors and a lamp actuator). The MEMOplanner supports the user with the calendar reminders and sensor-based reminders listed in Table 1 by issuing

[1] Abilia, HOMEbasic i2, http://www.abilia.com/sv/produkt/homebasic-i2?product_category=34 2015-11-09.

an auditory and visual reminder. Additionally, the light is automatically turned on when the user's feet are detected beside the bed to minimize the risk of falls during nighttime. In the current version, all information is saved locally within the MEMOplanner. While disabled by default upon delivery, a USB remote control can be provided to those who need to administer MEMOplanner. Provided that MEMOplanner is connected to the Internet, users (for example relatives or caregivers) with the USB can access all functionalities in MEMOplanner, i.e. also Windows. There is only one access level for remote administrators, hence, users provided with a USB can access and edit the content of the calendar, contacts, and photos from remote. A Vera 3 gateway communicates both with the sensors using the Z-wave protocol[2] and with the MEMOplanner (usually via WiFi but a cabled connection is possible). The Vera gateway can also connect the MEMOplanner to the Internet. By default, the triggered reminders are not stored within the Vera or sent remotely to any administrator.

Table 1. Summary of functionality of HOMEbasic

1.	*Calendar visualisation*: Information of time. Calendar for structuring daily tasks. The user (or a remote user) can add new events to the calendar. Each event can consist of sub events. It is also possible to add the relative's daily activities helping the person with memory decline keep track of his/her relatives' activities
2.	*Calendar reminders:* Reminders about upcoming events.
3.	*Additional functionality:* Skype, photo album, contacts
4.	*Sensor-based reminders:* Stove on but no motion in kitchen for x minutes. An electronic device has been turned on for x minutes. The entrance door is opened while the stove is on. The entrance door is opened. Can be used to remind the user to, e.g., lock the door or to tell the user that it is not an appropriate time to go out.
5.	*Sensor-based actuation:* Lamp automatically turned on when motion beside bed.

4 Research Method

This research was conducted as a case study where we investigated the CAT described in Sect. 3, as well as privacy concerns that arose in use of this technology in real homes. Although results from case study research cannot be statistically generalized, this approach supports collection of rich context-specific details and in this way reveals important information about the object under study and increases the understanding of the specific context [12]. By using this methodology, we were able to study the users' expectations and needs regarding privacy and the new generation of CAT's possibility to address these needs in depth.

[2] Abilia, MEMOplanner Handbook, fttp://www.abilia.com/sites/abilia.com/ files/field_resource_file/4X2650_Handbook_MEMOplanner_S525.pdf 2015-11-06.

4.1 Selection of Participants

We looked for test persons who had some sort of a cognitive decline. All seven test persons (μ = 71.6 years old) were men and lived together with their wives (μ = 65.7 years old) in ordinary housing/private residences outside nursing homes. All test persons and four of the wives were retired. Two wives were still working full-time and one was working part-time. Additional details about the case settings can be found in [13]. In sum, the CAT was deployed and used in seven Swedish homes during a period of approx. six months.

4.2 Data Collection

Data was collected in five stages: (1) functionality and privacy in design of HOMEbasic, (2) users' general privacy preferences, (3) observation of users' privacy preferences when using HOMEbasic, (4) users' privacy preferences in relation to HOMEbasic's current and future functionality (scenarios), and (5) users' opinions about the methods used for implementation and introduction of HOMEbasic.

Stage 1: functionality and privacy in design of HOMEbasic. In this stage, we wanted to gain a general understanding about HOMEbasic's functionality and the implemented privacy measures. For that reason, we participated in two demonstrations of HOMEbasic where we could interact with the system and ask questions. We also reviewed the HOMEbasic documentation and manuals and interviewed the developers. Finally, we interviewed experts responsible for prescription of this (and similar) CAT to end-users.

Stage 2: users' general privacy preferences. In this stage, data was collected through semi-structured interviews with the test persons and their relatives when the CAT was deployed. The interviews aimed to collect data about the participants' general requirements and preferences regarding privacy and their expectations/worries regarding privacy of information handled by the CAT. At this first interview session, the test persons and their relatives were interviewed separately. The interviews focused on three areas: (1) general privacy awareness and preferences, (2) privacy preferences in interaction with health care and elderly care, (3) privacy concerns in relation to use of the CAT. Aspects discussed within these areas were related to: the interviewees' privacy enhancing behaviors when using ICT, knowledge about privacy regulation, privacy awareness, privacy concerns etc.

Stage 3: observation of users' privacy preferences when using HOMEbasic's. During the test period, we visited the test sites regularly and observed how the CAT was used. We also discussed users' privacy concerns that come up during the use of CAT.

Stage 4: users' privacy preferences in relation to HOMEbasic's current and future functionality. This stage focused on privacy concerns in relation to present and future potential functionality of the CAT and was conducted when the CAT was removed from the homes. Future scenarios were developed based on the current trends in development of CATs aiming at integrating these technologies as a part of health care

Table 2. CAT's functionality and scenarios used during the interviews

Calendar visualisation.
Scenario 1: How would you feel if someone from your family or acquaintances studied the MEMOplanner with your daily tasks planned in it?
Scenario 2: Would you feel differently if it was a stranger e.g., someone helping with the renovation, who looked at your planned tasks? For instance, if you had planned a holiday?
Remote access (future scenarios)
Scenario 3: In this stage the test person gets help from home care staff and they can remotely access Memoplanner. A number of activities for both the test person and the relative are added in the MEMOplanner The test person is contacted by home care staff, pointing out that he exercises too little.
Scenario 4: The relatives is contacted by home care staff pointing out that she should encourage her husband to exercise more.
Calendar reminders.
Scenario 5: You have a party and there are a number of people in your home, acquaintances and family. The calendar reminds that [name of the test person] needs to take his medicine. The guests look at the calendar, some of them ask what it is. Some ask how [the test person] feels.
Scenario 6: Would you feel differently if it was a stranger, e.g., someone helping with the renovation who heard the reminder?
Additional functionality (future scenarios)
Scenario 7: You use the possibility to store photos in the MEMOplanner, i.e., you keep private photos of your family, photos from your vacation etc. After you come back from the holiday, home care staff comment on your holiday's photos which you have not shown them.
Scenario 8: Would you feel differently if they commented on pictures of your grandchildren from the last party?
Scenario 9: After a while you discover that your private pictures are used in an advertisement of the hotel on the Internet.
Sensor-based reminders (future scenarios)
Scenario 10: The stove sensor is installed and you feel that it fulfils its goal and runs the reminders when needed. On the next visit, your doctor tells you that your memory has declined because she/he can see that the reminders from the stove sensor runs more frequently.
Scenario 11: Would you feel differently if you knew that the doctor is able to see this information?
Scenario 12: Would you feel differently if it was a home care personnel who told you about the memory decline?
Scenario 13: The doctor points out that [the relative] should take more responsibility for cooking.
Sensor-based actuation (future scenarios)
Scenario 14: The motion sensor is installed in the bedroom. You feel that it works well and it feels safe when the lamp automatically turns on when you go up at night. You are contacted by the home care personnel who ask if you need any help with sleeping since they noticed that you go up often at nights.
Scenario 15: On the next visit, the doctor ask why you go up so often at nights. He/she suggests a medicine to help you sleep better.

and home care. For that reason, it is assumed in some scenarios that health care professionals and home care staff are able to access the data collected by the CAT. Use case-based scenarios steered the interview. We formulated at least one scenario for each CAT existing functionality in the CAT (see Table 2) and several possible scenarios for use of this functionality in the future. By default, the triggered alarms are not stored within the Vera or sent remotely to any administrator. However, accessing information about trigged alarms could allow home care staff and/or health care professionals to monitor changes in behaviors of the person using a CAT. For instance, more frequent actuation of the lamp sensor at nights could indicate that the elderly person has sleep problems or more frequently issued sensor-based reminders could indicate a decline of the elderly person's health condition. Such information could help the caregivers to react on the changes and help the elderly person in a more efficient way. Thus we included scenarios 10–15 assuming that it will be possible in the future.

For each scenario, the participants were asked a few questions revealing their privacy preferences, for instance: How does it make you feel? Why? What emotions does this event raise? Why? How would you like to change the situation to feel okay?

Stage 5: users' opinions about the methods used for implementation and introduction of HOMEbasic. This interview session, which was conducted one month after the CAT had been removed, focused on the test person's and the relative's reflections on the approach taken in this case study. The questions asked during these interviews were: What do you think about the information you got about the CAT? What do you think about the introduction and training? In relation to each question, we asked several follow up question such as: Was it enough/not enough? How would you like to get the information/training? What was missing?

4.3 Data Analysis

The collected data was analyzed in 4 steps. First, we identified the privacy preferences highlighted by the users in relation to the CAT's functionality during all stages of data collection. Second, we identified the privacy implementations from the material collected during the first stage of data collection in order to find how PbD principles were incorporated into the design of HOMEbasic. Third, we identified the users' unsolved privacy requirements in relation to each of these principles. Finally, based on the analysis and current literature, we formulated guidelines for applying PbD principles in design of the new generations of CAT.

5 CAT Users' Privacy Requirements

In this section we describe users' privacy requirements in relation to the current and possible, future functionality of HOMEbasic. The section is structured according to the functionality categories presented in Table 1. For clarity, illustrative examples in relation to each principle are provided.

5.1 Calendar Visualization

Some privacy concerns regarding this functionality came up already during the second stage of data collection (see Sect. 4.2), when we asked about users' privacy concerns in relation to HOMEbasic before they started to use it. Mainly the relatives expressed privacy concerns in this stage, while the test persons were less worried about privacy violation when using HOMEbasic. For instance, a relative in the test site that still had a teenage child at home was very concerned about the child's privacy and did not want to add any events that would reveal information about the child into the calendar even if this information would be helpful for the father who suffered from a strong cognitive decline. Another wish that came up during this stage of data collection was the possibility to delete "old" data stored in the MEMOplanner. This requirement was expressed by both the test persons and their relatives. Users preferred the data to be deleted frequently by default but if it was not possible they wanted to be reminded to do it by themselves. The users informed that the MEMOplanner may store detailed descriptions about how the activities should be performed, including sensitive information about the test person's health condition and needs. An interesting reflection that was made by one of the relatives during this stage of data collection was that privacy requirements change in line with progression of the disorder. The respondent explained: *I think that when you get to that stage when you can no longer cope with things alone and need home care to help you with everything, the privacy is already forgotten. It is a sad part of it, but so it is.* This means that privacy requirements are not static and that the new generation of CAT should be able to handle these changes.

During this stage of data collection, two relatives also mentioned a need for limiting access to the information visible in the MEMOplanner. This need becomes more apparent during stage 3 and 4 of the data collection. For instance during stage 3, we could observe that one of the wives kept deleting events that already occurred from the calendar. Asking her about the reason for doing it, she answered: *We could have done something in the morning, then maybe we had guests in the afternoon or evening, and then I thought that they did not need to know what we have done in the morning.*

When discussing scenarios 1–2 in stage 4 of the data collection, we found that both the test persons and the relatives are concerned about possible privacy violations when the information in the MEMOplanner is visible for strangers who for different reasons are present in the elderly peoples' home. One of our respondents told us: *I would not like it if a stranger looked at our planning [in the MEMOplanner]. I would feel uncomfortable. In this case, I would need to turn the calendar off, or somehow make information invisible. For now, the calendar is completely open.* Another respondent explained: *The idea is to put everything in the MEMOplanner, e.g., that you have to visit a special doctor. You may not want everyone to know about this and about what you do during the days. I think it depends on family relationships and how you are as a person. Some people do not want others to know anything….* Thus, we identified a clear need for a possibility to sometimes hide the information in the MEMOplanner.

5.2 Calendar Reminders

The users' privacy concerns related to this functionality were identified mainly in stage 3 and 4 of the data collection (see Sect. 4.2). For instance, while visiting a test site, we noticed that the volume of the reminders was heavily lowered. As a consequence, the test person could hardly hear the voice reminders. When we asked them why they lowered the volume for the reminders, the wife explained that she felt embarrassed when the neighbors could hear the voice reminders so she decided to lower the volume. Additional clear privacy preferences were formulated by the users during stage 4 when we discussed scenarios 5 and 6. We found that the users had different privacy preferences regarding this functionality. While some users did not mind if other people heared the reminders, other users were clearly uncomfortable with this. One of our test persons told us: *I don't feel comfortable with this. It makes you feel sick! I try not to think about my disorder all the time. In this case, I would get lots of questions that I do not want to get or discuss. It's a party and I also want to have fun. I would like the MEMOplanner or the reminders to be switched off just then.*

5.3 Additional Functionality and Remote Access

Privacy concerns related to this, future possible way of using MEMOplanner were revealed during stage 4 when we discussed scenarios 3–4 and 7–9. The current version of the MEMOplanner offers the possibility to store contacts and photos and to use Skype for communication. In the future scenarios, we assumed that home care personnel will have remote access to the MEMOplanner to be able to help the user to plan the activities. In the current version of the CAT, all information is equally accessible for all users and there is no possibility to restrict access to certain parts of the stored data. However, while discussing scenarios 3–4 and 7–9, we found that photos and in some cases contacts are considered as sensitive for some users and for that reason they should not be accessible for all categories of current or future users. One test person told us: *I would not like it if the homecare staff looked at my photos without permission. This is not a part of their job! Maybe they would not do it, but because I cannot prevent it, I cannot be sure.* Thus, the users emphasized the importance of access control mechanisms allowing them to decide who is permitted to access specific information stored in the MEMOplanner (e.g., planning, details regarding each planned event, photos etc.). One of the users said: *you do not want everyone to know what information you put in [the calendar]. Then you can get worried about how that information is disseminated. You should be able to control who sees what. As it is in other systems; some people can access the information and others cannot.*

5.4 Sensor-Based Reminders and Actuation

Privacy concerns in relation to the future use of these functionalities were discussed in stage 4 of the data collection using scenarios 10–15. We found that most test persons would welcome such functionality if the main reason for it was to help them in their disorder. However, there was a clear difference in the users' privacy concerns

depending on which group of caregivers that would monitor the sensor-based reminders and actuations. The users did not have any restrictions in the case of health care professionals. They did not even care if they would be informed about the monitoring in advance or not. The users were more restrictive in the case of homecare staff. They could accept the monitoring if they were informed in advance about the purpose and extent of the monitoring. They also highlighted it as important to limit the number of home care staff (preferably only one contact person) that could access the log data. Generally, they felt that monitoring could increase their sense of safety at home, when they could no longer cope with the basic things by themselves.

Table 3. Summary of the users' privacy requirnements in relation to CAT's functionality

Calendar visualisation.
R 1: a possibility to delete "old" data stored in the CAT (by default)
R 2: new generation of CAT needs to handle changes of users' privacy needs
R 3: a need for a possibility to sometimes hide the information visible in the CAT
Calendar reminders.
R 4: a need for a possibility to sometimes switch off voice reminders
Additional functionality and remote control
R5: a need for access control mechanisms allowing the users to decide who is permitted to access specific information stored in the MEMOplanner
Sensor-based reminders and actualisations
R 6: a need to differentiate the access to the collected sensor data dependently of the caregivers
R7: a need for knowing in advance about the purpose and extent of the monitoring.
R 8: a need to limit the number of home care staff (preferably only one contact person) that could access the log data

Table 3 summarizes the users' privacy requirements (R) in relation to CAT's functionality. Some of the requirements were clearly stated by the users, others were derived by the researchers based on users' statements.

6 Guidelines for Applying PbD Principles in Design of CAT

In this section, we discuss findings from the case study in relation to the PbD principles formulated by Cavoukian et al. [8] and existing literature. Based on the discussion, we formulate a set of guidelines for applying PbD principles in design and use of the new generation of CAT.

6.1 Principle 1: Proactive not Reactive; Preventative not Remedial

The importance of privacy protection is not emphasized in the documentation of the CAT which focuses on the description of functionality allowing users to live safely and independently in their homes. Based on the documentation, we can conclude that the design of HOMEbasic is safety- and functionality-driven and users' privacy is not especially emphasized. However the users highlighted many privacy concerns in relation to the existing and future possible use of CAT (see Sect. 5). They also expressed several privacy requirements in relation to this technology (see Table 3). Following the first PbD principle means to design CATs with these privacy requirements in mind to be able to prevent privacy invasive events before they happen. Therefore the identified privacy requirements should be considered when CAT are designed. Another important finding from our case study is that different users experience different events as privacy invasive and that contexts in which CAT are used are very different. The problem of not addressing the diversity of elderly users is highlighted in literature. The elderly users are often treated as a homogenous group with similar needs and preferences [5, 11]. But to be able to truly meet elderly peoples' needs and preferences, the designers need to acknowledge and understand the differences [6]. *Thus, the designers of the new generation of CAT should consider the variety of users and contexts in which CAT can be used and design adaptable privacy solutions that prevent the privacy invasive events to happen (Guideline 1).* By adaptable solutions we mean flexible solutions that are possible to adapt to the diverse privacy requirements and diverse context of the elderly users homes.

6.2 Principle 2: Privacy as the Default

We found that privacy is built into HOMEbasic to some degree. For instance, the product's documentation states that the possibility to remotely access the MEMOplanner is disabled by default and must be turned on manually because of privacy concerns. In this way, the data stored in the MEMOplanner is protected against unauthorized usage by default and the users do not need to take any additional action to protect the sensitive data in the default setup.

However, we discovered several examples (see Sect. 5) when protection of privacy required users to actively take necessary actions, otherwise there was a risk for privacy violation. In some cases, the desired level of protection was not even achievable because of lacking technical implementations, i.e., access control and screen saver. We found that our participants would like the CAT to protect their privacy by default. Protecting privacy by default may result in inflexible and unadaptable solutions that cannot handle changing privacy requirements. As described in Sect. 5.1, the elderly person's health situation can change over time and this can lead to changed privacy preferences. Treating privacy requirements as static is considered as being problematic in the literature [5, 14]. *Thus, the designers of the new generation of CAT should investigate users' privacy preferences to find situations when privacy protection can be built into the technology by default. Default settings should be balanced with flexibility (Guideline 2).*

6.3 Principle 3: Privacy Embedded into Design

Regarding the third principle, we found that users would like to have additional technical solution to protect their privacy. It is argued in literature that most people are not capable of protecting their own sensitive information [9], thus embedding privacy in technical solutions is important. However it is also argued that solely technical solutions cannot solve the complex privacy challenges that arise when using the new generation of CAT [5]. *Thus, designers of new generation of CAT should provide users with privacy guidelines and recommendations on how to protect privacy when the CAT is in use. Such guidance is missing in the CAT's documentation and manuals. (Guideline 3)*

6.4 Principle 4: Functionality—Positive-Sum, not Zero-Sum

Our findings in relation to principle 4 indicate that it is not easy to balance utility and privacy in use of the new generation of CAT. In the provided examples (see Sect. 5), we can see that privacy measures such as hiding information in the MEMOplanner or lowering the volume of the reminders may lead to a decreased possibility to support the person affected by cognitive decline (see Sects. 5.1 and 5.2). The conflict between privacy and other values such as safety and autonomy is well known in literature and often highlighted as problematic [9]. *Thus, designers of the new generation of CAT should perform a privacy and utility analysis when a new functionality is added to the system (Guideline 4). There are situations where new functionality does not contribute to utility and jeopardize privacy*. Such development should be avoided.

6.5 Principle 5: End-to-End Lifecycle Protection

Although we studied only a small sample of seven test sites, we could observe differences in privacy preferences that depended on contextual variables such as family situation, family relationships, how active the elderly people were and how many people who visited their homes. We argue that these different privacy needs would not be identified if we did not studied the use of CAT in real settings. Thus, we can conclude that it is important to study CAT in use to be able to implement privacy measures that are relevant and adapted to the specific needs of the different users. The problem of a lack of real experiences of using CAT in practice and thus a lack of knowledge about possible privacy concerns that can arise when using such technology is highlighted in literature as problematic [3, 11]. *We argue that designers of CAT should consider privacy aspects regarding the use of CAT in real settings already during the development process (Guideline 5)*. It can be done by involving significant stakeholders, such as primary users, secondary users, health care professionals, and other formal and informal caregivers in design of CAT.

6.6 Principle 6: Visibility and Transparency

To comply with this PbD principle is a challenge in the CAT context because often elderly people have difficulties in understanding the consequences that the implemented

technology have on their privacy. The problem is also highlighted in literature. For instance Bowes et al. [11] argue that elderly people do not have the necessary (technical) background to formulate the appropriate privacy requirements and understand the privacy consequences of the implemented CAT. Although most of the participants told us (see 4.2, stage 5) that they experienced the CAT to be complex and sometimes difficult to use, they were able to discuss privacy issues in relation to it. The understanding came after they had used the CAT for some time in their homes and because we used methods and tolls to exemplify and visualize current and future use of the CAT i.e. scenarios, audio/video presentation. *Thus we argue that it is important to use specific methods and tools to make the privacy consequences understandable and clear for the elderly people who will use the new generation of CAT (Guideline 6).*

6.7 Principle 7: Respect for the Users' Privacy

Regarding the seventh principle, we can conclude that elderly people are both capable and willing to engage in discussions about their privacy preferences. We also found (see Sect. 4.2, stage 2) that elderly people do care about privacy. Generally, they are careful about how they reveal their personal information in use of technology, Internet, social media and even in their everyday life. They are mostly concerned that their personal information may be used by criminals. Regarding revealing health information, they totally trust health care professionals and are not concerned about providing such information to health care organizations. Regarding privacy in relation to information handled by CAT, we can conclude that the preferences differed between the users' depending on personal preferences, environmental factors, and family relationships. *Thus, we argue that designers of CAT should involve the elderly users in the design of the new generation of CAT and the design of privacy solutions in relation to this technology (Guideline 7).*

7 Conclusion

The rapid development of cognitive assistive technologies (CAT) from simple analogue assistive devices into complex and sophisticated sensor networks raises many new privacy issues that are not sufficiently addressed in design and use of these technologies in real homes. This paper investigated how the new generation of CAT incorporates PbD principles suggested by Cavoukian et al. [8]. Special attention was put on that users' privacy preferences that arise in use of new generation of CAT in real homes. Based on our empirical findings and a literature review, we suggest a set of guidelines for applying Cavoukian et al.'s PbD principles in design and use of the new generation of CAT. Using these guidelines will help to build in users' privacy requirements in the design of a new generation of CAT and in this way protect the privacy of the elderly people using these technologies.

Generally, we can conclude that the existing PbD principles focus on technical aspects of privacy and often reduce privacy to the protection of personal data. Our examples (see Sect. 5) show that some privacy concerns arising in the use of CAT are

beyond this narrow view of privacy. For instance, we found that privacy in this context is not only related to the primary user of the CAT but also to people who are in the user's environment such as a child or wife etc. We argue that more research of non-technical privacy aspects of privacy in this context is needed.

Acknowledgment. The authors would like to thank the elderly participants and the Länsförsäkringar Research Foundation for making this study possible.

References

1. Frennert, S.A., Forsberg, A., Östlund, B.: Elderly people's perception of a telehealthcare system: relative advantage, compatibility, complexity and observability. J. Technol. Hum. Serv. **31**, 218–237 (2013)
2. Koch, S., Marschollek, M., Wolf, K.H., Plischke, M., Haux, R.: On health-enabling and ambient-assistive technologies. What has been achieved and where do we have to go? Methods Inf. Med. **48**, 29–37 (2009)
3. Kosta, E., Pitkänen, O., Niemelä, M., Kaasinen, E.: Mobile-centric ambient intelligence in Health- and Homecare-anticipating ethical and legal challenge. Sci. Eng. Ethics **16**, 303–323 (2010)
4. Shankar, K., Camp, L.J., Connelly, K., Huber, L.: Aging, privacy, and home-based computing: developing a design framework. IEEE Pervasive Comput. **11**, 46–54 (2012)
5. Zwijsen, S.A., Niemeijer, A.R., Hertogh, C.M.P.M.: Ethics of using assistive technology in the care for community-dwelling elderly people: An overview of the literature. Aging Mental Health **15**, 419–427 (2011)
6. Frennert, S.A., Östlund, B.: Review: seven matters of concern of social robots and older people. Int. J. Soc. Robot. **6**, 299–310 (2014)
7. European Commission: Proposal for a regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation) (2012)
8. Cavoukian, A., Fisher, A., Killen, S., Hoffman, D.: Remote home health care technologies: how to ensure privacy? Build it in: Privacy by design. Identity Inf. Soc. **3**, 363–378 (2010)
9. Nordgren, A.: Privacy by design in personal health monitoring. Health Care Anal. **23**, 148–164 (2013)
10. Mort, M., Roberts, C., Pols, J., Domenech, M., Moser, I.: Ethical implications of home telecare for older people: a framework derived from a multisited participative study. Health Expect. **18**, 438–449 (2015)
11. Bowes, A., Dawson, A., Bell, D.: Implications of lifestyle monitoring data in ageing research. Inf. Commun. Soc. **15**, 5–22 (2012)
12. Yin, R.K.: Case Study Research-Design and Methods. SAGE Publications, Thousand Oaks (1994)
13. Kristoffersson, A., Kolkowska, E., Loutfi, A.: Assessment of expectations and needs of a sensor network to promote elderly's sense of safety and security. In: The Seventh International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services, CENTRIC 2014, pp. 22–28 (2014)
14. Remmers, H.: Environments for ageing, assistive technology and self-determination: ethical perspectives. Inf. Health Soc. Care **35**, 200–210 (2010)

A Trustless Privacy-Preserving Reputation System

Alexander Schaub[1(✉)], Rémi Bazin[1], Omar Hasan[2], and Lionel Brunie[2]

[1] Ecole Polytechnique, 91128 Palaiseau, France
alexander.schaub@polytechnique.edu
[2] University of Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, 69621 Lyon, France

Abstract. Reputation systems are crucial for distributed applications in which users have to be made accountable for their actions, such as e-commerce websites. However, existing systems often disclose the identity of the raters, which might deter honest users from submitting reviews out of fear of retaliation from the ratees. While many privacy-preserving reputation systems have been proposed, we observe that none of them is simultaneously truly decentralized, trustless, and suitable for real world usage in, for example, e-commerce applications. In this paper, we present a blockchain based decentralized privacy-preserving reputation system. We demonstrate that our system provides correctness and security while eliminating the need for users to trust any third parties or even fellow users.

1 Introduction

These days, reputation systems are implemented in various websites, where they are crucial for the customer experience. For instance, buyers are inclined to pay more for goods if the seller has a good reputation [1]. One of the first and best-studied systems in the e-commerce domain is the reputation system at ebay.com [2]. Its main objective is to help prospective customers to determine the trustworthiness of the sellers, and thus minimize the risk of fraud.

A study [2] showed that users may retaliate in case of negative feedback, and thus raters are less likely to provide honest feedback. In order to avoid this problem, several privacy preserving solutions have been proposed. Some of them try to hide the identity of the ratee [3–5], while others try to hide the rating [6–8] while making the aggregated reputation public.

While some of the existing privacy preserving reputation systems might be suitable for e-commerce applications, we observe that each one of them comes with its drawbacks. For example, Kerschbaum's system [9] has been specifically designed with e-commerce in mind. However, it is a centralized system, and thus can potentially be abused by the central authority. Other schemes [8] achieve anonymity even in this context, but are not trustless.

Given these considerations, we would like to achieve a *trustless reputation system*, i.e. one that does not require the participants to trust other users or entities to not disrupt the protocol or to breach their privacy. This privacy-preserving

J.-H. Hoepman and S. Katzenbeisser (Eds.): SEC 2016, IFIP AICT 471, pp. 398–411, 2016.
DOI: 10.1007/978-3-319-33630-5_27

reputation model should be suitable for e-commerce applications, and we will therefore suppose that the identity of the customer is revealed during the transactions that they can rate.

In order to achieve true trustlessness we also require our system to be decentralized. One way to obtain decentralization is to use a distributed database in order to store the ratings submitted by the customers. We will achieve this using *blockchains*.

The blockchain technology, which became popular thanks to the BitCoin protocol [10], has been used in various applications. Among these applications, we can count a domain name system (DNS) named Namecoin. The blockchain can be more generally, as explained in [11], seen as a public distributed database, with all the participants agreeing about its state in a secure manner. In BitCoin, for example, this database serves to store a ledger of the coins that each user owns, as well as the transactions between the users.

Anonymous reputation systems are a natural application for the blockchain technology. There have already been some attempts at building such systems [12], however, there seems to be no usable solution yet.

We will leverage this technology in order to achieve the objectives of our reputation system. It will enable us to build a truly decentralized system, that does not require the participants to trust other users, as the integrity of the rating-history can be verified by every user.

We propose a truly trustless, decentralized, anonymity preserving reputation system that is suitable for e-commerce applications. It is based on the blockchain technology, and will induce low overhead for the processing of transactions, while at the same time be robust and allow customers to submit ratings as well as textual reviews.

The rest of the paper is organized as follows. In Sect. 2, we will analyze existing privacy-preserving systems and explain in further detail why they are not suitable for e-commerce applications. In Sect. 3, we will explain the model used for our system, and list the properties that we want to achieve. Then, in Sect. 4, we will describe the necessary building blocks, and in Sect. 5, we will present our system in detail. Finally, we will explain in Sect. 6, why this system meets the expected goals. We conclude the paper in Sect. 7.

2 Related Work

Privacy preserving reputation systems have been studied in the literature for a long time. One of the first proposed systems was designed by Pavlov et al. [6] and uses primitives such as the secure sum and verifiable secret sharing. It protects the confidentiality of the feedback by hiding the values of the submitted ratings. Hasan et al. [8] later introduced a system based on additive homomorphic cryptography and Zero-Knowledge proofs where the privacy of a given user can be preserved even in the presence of a large majority of malicious users. A little later, Dimitriou et al. [7] proposed two protocols with a similar architecture to the systems presented by Hasan et al., with slightly higher

asymptotic complexity, however, less demanding in terms of resources for the querier (he has to relay less messages, verify less proofs, etc.).

Some protocols [6–8,13,14] are truly decentralized and the feedback is retrieved from the participants every time a querier wishes to learn the reputation of another participant. Therefore, all the nodes have to stay online in order to contribute to the reputation system, which is not suitable for e-commerce applications, but might be useful in other contexts, such as P2P applications.

Hence, we will focus on privacy-preserving methods that completely hide the identity of the raters. Protocols of such type do already exist, however each one of them has its own weaknesses. The works of Androulaki et al. [13] and Petrlic et al. [14], for example, are instances of pseudonym based schemes. Nonetheless, these two require a Trusted Third Party (TTP), and are thus not truly decentralized. As the TTP has to be completely trusted for certain operations, its misbehavior could breach the privacy of the users or the correctness of the system.

Anceaume et al. [3,4] proposed slightly different solutions. Instead of all the information about the reputation of the users being held by a single TTP, they distribute the trust using a DHT-structure: every peer holds some part of the information, which allows to compute the reputation of a service provider. Moreover, in their system, peers rate *transactions* between customers and service providers, rather than directly rating service providers. This seems more suitable for e-commerce applications, as one would typically rate every transaction made with a service provider, rather than periodically update their opinion on a given service provider. It also allows to introduce proofs of transactions, which guarantee (more or less) that only transactions that really took place can be rated. However, as the service provider creates those proofs, it is complicated to ensure that he doesn't generate proofs for transactions that did not happen, in order to submit positive reviews by himself and wrongfully increase his own reputation. Anceaume's and Lajoie-Mazenc's systems only offer little protection against these attacks. The system proposed in [4] also makes use of complicated zero-knowledge proofs and is thus quite costly to perform (several seconds for each participant, up to a minute in certain cases).

None of those protocols are trustless, and therefore need either the customers or the service providers (or both) to trust some entities not to tamper with the system or to break privacy, without being able to verify that there is no bad behavior. We eliminate this weakness in the system that we present in this paper.

3 Our Model

3.1 Participants

For our system, we choose a model that is as close as possible to actual e-commerce systems. As stated in Sect. 1, we will consider two types of users: service providers (SP) who will sell goods or services, and customers who might buy them. The most important part in e-commerce rating systems is the rating of the service providers (as opposed to ratings from the seller about the buyer).

Therefore, we will only consider ratings *from* the customers *about* the SPs. Only customers might be raters, and only SP will be ratees.

We will also suppose that the transaction will ***disclose*** the identity of the customer: the SP will need the customer's credentials, such as his credit card number or address, in order to process the order. Even if the transaction is done via an anonymous electronic currency such as Dashcoin or Zerocash, the service provider will most certainly need the customer's address in order to deliver the good. We suppose that after every transaction between a customer and a SP, the customer might rate the SP.

More formally, we will introduce the following notations:

S: The set of all the service providers (i.e. ratees)

C: The set of all the customers (i.e. raters)

P: The set of all the participants, $P := S \cup C$. It is simply the set of all the nodes participating in the network.

B: The blockchain.

As the blockchain defines an ordered set of blocks. A block is simply a set of operations that are aggregated for maintenance reasons (it is more efficient to store them this way). The blockchain can also be seen as a database whose state will be the initial state (that is hard-coded) on which all the operations contained in the subsequent blocks are applied.

Every time a new block is constituted, an award will be paid to the user that constituted it. This works in a similar fashion as in the so-called "alt-coins". In our system, owning coins is mandatory in order to be allowed to receive reputation. It also helps preventing spam and other kinds of attacks (as described in Sect. 6.2).

A: The set of all the addresses of the participants.

These addresses will be used for maintenance. Every service provider will own one address. They will be used, in particular, to hold and spend the coins generated by the blockchain, but also to identify the service providers.

Service providers will have a unique address, as issuing reputation tokens will cost coins and owning an address is necessary in order to own and transfer them. As a service provider will not gain anything from having more than one address (see Sect. 6.2 for more details), there is no need to try and enforce this policy. Furthermore, it would be complex to enforce it in a decentralized fashion.

3.2 Operations

We will next describe the functions that are needed in our system. The protocols that implement these functions will be described in the later sections. Most, if not all, of these functions will be performed with respect to a given blockchain B, or need to make calls to a random number generator (RNG). These are implicit inputs of the protocols.

For the Customer. These operations will be performed by a certain customer $c \in C$.

- setup()
  Generates a new public key, usable by the customer, for the transaction.
- get_reputation(s)
  Allows the customer to query the reputation of a service provider $s \in S$.
- get_token(s, x)
  Allows the customer to request a token that will prove that he was engaged in a transaction x with the service provider s. Outputs a blinded token $\bar{t}_x$.
- unblind_token($\bar{t}_x$)
  Unblinds the token that was retrieved using the `get_token` protocol. Outputs an unblinded token t_x. The token is bound to the transaction. However, given two tokens t_x and $t_{x'}$, the service provider s will not be able to tell which token belongs to which transaction.
- publish_review(s, t_x)
  Allows the customer to publish a review about the service provider $s \in S$, using the token t_x previously unblinded.

For the Service Provider. This operation will be performed by the service provider.

- issue_token(c, x)
  In response to a `get_token` request from the customer $c \in C$, issues a blinded token $\bar{t}$ and sends it to the customer, if the customer is entitled to receive one, i.e. he was really engaged in the transaction x.

Block-Chain Related Operations. These operations are mostly independent from the underlying reputation model. However, a blockchain mechanism is needed in order to store the reputation values in a reliable way. These operations can be performed by any node in the network.

- broadcast(op)
  Broadcasts an operation op (which can be a review, a transaction, etc.) to all the nodes running the protocol.
- compute_balance(s)
  For a service provider $s \in S$, representing an address $addr$, computes the balance (in terms of coins) associated with this service provider.
- create_new_block($addr, b$)
  Broadcasts a newly mined block b. It will contain, among other data, reviews and transactions, but also a proof that $addr$ has the right to constitute the next block.

The incentive for creating blocks are the *coins*. Every address who correctly creates a block and broadcasts it will receive a certain amount of coins. They serve as a currency within this system, in a similar fashion as in many cryptocurrencies that have been developed since BitCoin. However, in our system,

the main usage of the coins is *not* to serve as an alternative currency. Rather, they will be needed by the service providers in order to have the right to deliver tokens. This also serves to limit spam from the service providers, who could simply create as many tokens as they desire in order to boost their reputation.

3.3 Adversarial Model

We consider a malicious adversarial model with collusions. This model implies that any participant in the protocol may behave arbitrarily and deviate from the protocol at any time as deemed necessary. Service providers may want to learn the identity of the customers that rated them, they might try to raise their own reputation, and collaborate with other service providers. Customers may try to submit reviews without having previously interacted with service providers, might try to use the received token in order to rate other service providers, or might try to otherwise disrupt the service. We will also suppose that there might be attempts to disrupt the blockchain, such as forking in order to confuse new participants.

3.4 Objectives

The objectives for our system are the following:

- *Trustlessness*
  In an e-commerce system, we cannot expect customers to have pre-existing trust towards other customers of the same SP. Therefore, our system should not suppose that there is pre-existing subjective trust between users. No customer should have to trust any entity *not* to deviate from the protocol in order to break its privacy or change its rating. The protocol should ensure that privacy and correctness are preserved even if other participants deviate from the protocol. Therefore, it should also not rely on Trusted Third Parties, or Certification Authorities, which, by definition, must be trusted to behave faithfully.
- *Suitability for e-commerce*
  As the identity of a customer will be most certainly revealed during a transaction, the system should enforce the unlinkability of transactions and ratings, i.e. for a given rating, it should not be possible to determine which transaction it is related to (it should however be possible to identify the related SP). It should, however, not be possible for a customer to submit a rating if no transaction took place.
- *Decentralization*
  We want to avoid any central point of failure as well as any single point of control. Therefore, the system should not depend on one, two, or a small number of nodes in order to work properly. We will even exclude Certification Authorities, because they have proven unreliable in the past, either because they became subject to attacks [15,16] or because they issued themselves fraudulent certificates [17], and because they would induce some centralization aspects in the system.

– *Anonymity preservation*
  The anonymity of the customers should be preserved. More precisely, the ratings and the identities of the customers should be unlinkable, as well as the ratings among themselves. The later kind of unlinkability is also crucial to preserve the anonymity of the users, as highlighted in [18,19].
– *Robustness*
  Our system should be robust to classical attacks on reputation systems, in particular bad-mouthing, ballot-stuffing, Sybil attacks and whitewashing.

4 Building Blocks

In order to be able to build this protocol, we will need two basic building blocks: the blockchain and blind signatures.

4.1 Blockchain

A blockchain, as first desribed in [10], can be seen as a distributed, public database, which can be read by every user running the appropriate program, but on which writing has a cost, or cannot be done at any time by any user [11]. Every action that modifies this database is broadcasted among all the users in the network, and they are recorded as "blocks". The creation of those "blocks" is controlled by mechanisms that vary between the different blockchain algorithms, and the state of the database is the sum of all the actions in all the blocks at a given moment in time. This concept has become popular due to the BitCoin currency [10], which seems to be the first application making use of this idea. Two families of blockchain systems have since then emerged.

The first one uses a mechanism for controlling the blockchain that is similar to the one used in BitCoin, in which the probability of a participant creating a new block is proportional to its computing power. The second one uses a different mechanism, in which the amounts of coins held by the participant define this probability. This is called Proof-of-Stake, and we advocate the use of such a blockchain system for our protocol. More information about the different blockchain systems can be found in the extended version of this paper [20].

4.2 Blind Signatures

A blind signature scheme is a protocol in which the signer of a message does not learn anything about the content of the message that was signed. We expect from such a system the following properties:

Unforgeablility
  The signature cannot be falsified (only the user knowing some secret information, such as a private key, can issue valid signatures).
Blindness
  The signer does not learn anything about the message it signs (given the information available to the signer, all possible messages are equally likely to be about to be signed).

Untraceability
Once the message and signature have been revealed, the signer cannot determine when the message was signed.

For example, the blind signature scheme proposed by Okamoto [21], based on bilinear pairings, could be used to instantiate this primitive, or the simpler version based on the RSA algorithm, first proposed by Chaum [22]. As Chaum's version is simpler and faster, we will use this scheme in order to explain our protocol.

5 Specification of the Protocol

5.1 An Overview

The proposed protocol could be summarized as follows:

1. Before contacting the service provider in order to perform a transaction, the customer may compute the service provider's reputation using the get_reputation protocol.
2. Once the customer retrieved the reputation of the service provider, he decides whether to engage in a transaction with the SP or not.
3. If the customer decides to engage in a transaction, before a transaction takes place, the customer creates a new *public key*, derived from a private/public key pair, for the process. This key should be kept secret from the service provider for the moment. It will be used to avoid token-theft Then, the transaction takes place: for example, the customer sends the money, and the SP starts to deliver the good.
4. Just after the transaction takes place, the customer asks the SP for a *blinded token* by performing the get_token protocol (which takes the freshly generated public key as input), and verifies that the SP has a sufficient balance for issuing a token (using the compute_balance protocol). The balance should be greater than some n coins, since n coins will be deduced from the SP's account when the review will be integrated in the blockchain. The customer then verifies the token (i.e. verifies that the signature is correct) and unblinds it for later use with help of the unblind_token protocol.

   Requiring some coins to be spent in order to receive a review helps to prevent ballot-stuffing attacks, as the SPs may, theoretically, issue an unlimited amount of tokens to themselves and could therefore submit an unlimited number of positive reviews for themselves. As for the token, it serves as a proof that a transaction really occurred. It therefore helps to greatly reduce the risk of bad mouthing attacks. It has to be blinded, so that the service provider cannot link the token, and therefore the rating, to the transaction and the identity of the customer.
5. Once the customer is ready to review the SP, he will broadcast a message containing the address of the SP, the token, along with the rating of the transaction and (optionally) a written review, a signature on this information, as well as a pointer to the last review concerning the same service provider. This is done via the publish_review protocol. a cash system.

6. A participant wishing to earn coins (in order to be allowed to grant tokens for example) verifies if he is allowed to constitute the next block. If it is the case, he will run the constitute_block protocol. The creation of blocks helps to maintain a unique history of actions, avoids double-use of tokens, and is incentivized through the reward in coins.

In the next sub-sections, we will describe the protocol in more detail.

5.2 Public Key Creation

Before the transaction takes place, the customer creates a new public key that will be used for one transaction only (similar to what is recommended for BitCoin addresses for example). This will be the public part of an ECDSA key [23]. It must not be communicated to the SP during the setup phase. An outline of the setup protocol could look like follows:

Algorithm 1. Setup protocol

```
1: procedure SETUP(T)
2:     (p, a, b, G, n, h) ← T // Those are the parameters of the elliptic curve used for
   ECDSA, for example those of secp256k1
3:     privKey ← rand(0, n)
4:     pubKey ← privKey ∗ G
       return (pubKey, privKey)
```

5.3 Blinded Token Exchange

Before the transaction takes place, the customer will receive a token from the SP that will guarantee that its review will be accepted. For this purpose, the customer hashes the previously generated public key and requests a blind signature on this, for example using Okamoto's provable blind signature scheme (in the complete, not partial blinding setup), or the much simpler Chaum's blind signature algorithm. This will make the token unlinkable to the transaction, and therefore guarantee the anonymity. The customer will also check that there are enough coins in the wallet associated with the SP. Then, the transaction can take place.

If Chaum's blind signature is used, then we can define the three protocols get_token(s), unblind_token($\bar{t}$) and issue_token(c) as in **Algorithm** 2.

We suppose that the customer knows the public key of the SP. How this is done is out of scope of this paper.

Algorithm 2. Token exchange

procedure GET_TOKEN($s, pubKey, e, n, x$) ▷ (e, n) is the service provider's public RSA key pair, x the identifier of the transaction
  $m_1 \leftarrow hash(pubKey)$ ▷ $hash$ is a cryptographic hash function such as sha256
  $r \leftarrow rand(0, n)$
  $m_1 \leftarrow m_1 r^e \bmod n$
  $send((m_1, x), s)$
  return (m_1, r)
procedure ISSUE_TOKEN(c, m_1, d, n, x) ▷ (n, d) is the service provider's private RSA key pair
  if verify(x) **then** ▷ the service provider has to specify verify
    $\bar{t} \leftarrow m_1^d \bmod n$
    $send(\bar{t}, c)$
    return $\bar{t}$
procedure UNBLIND_TOKEN($\bar{t}, r, e, n$)
  $t \leftarrow \bar{t} r^{-1} \bmod n$
  return t

5.4 Broadcasting the Review

Once the transaction is finished, the customer might want to wait for some time (so that he is not the SP's only customer for this period). After this period of waiting, he might choose a rating for this transaction (say, an integer in $[|0; 5|]$) and write a review about it. The review can give helpful information to prospective customers, explain a bad rating, and helps distinguishing between trustworthy and fake ratings. This information will be broadcast in the network, along with the identifier of the SP, the token and the signature on the token. This message will also contain the signature of the customer, and a pointer to the last review concerning the service provider. The message that will be broadcast can be represented as follows (Table 1):

Table 1. Structure of a broadcasted message containing a review

Field	Description
$addr_s$	Address of the SP
$pubKey$	The public key used for the transaction
$token$	Token obtained from the SP (i.e. blind signature on $pubKey$)
$rating$	Rating of the transaction (for example, an integer in $[0; 5]$)
$review$	A textual review on the transaction (optional)
sig	Signature, using $privKey$, of $(adds_s \| \| pubKey \| \| token \| \| rating \| \| review)$
$pointer$	Pointer to the last review about the same SP

5.5 Computing the Reputation

In order to compute the reputation, a new customer only needs the last block containing a review about the SP whose reputation it seeks. Once this block has been found, it is sufficient to follow the pointers in order to retrieve all the reviews about this SP. For each review, the prospective customer might also verify the correctness of the blinded tokens. Then, the customer can choose any aggregation function he wishes (mean, median, or beta-reputation [24]), and could also read the textual reviews in order to filter out outlier ratings (especially high or, more probably, especially low ratings).

6 Analysis of the Protocol

6.1 Security Analysis

In this section, we list the theorems whose proof demonstrates that we achieve the security objectives of our protocol. The proofs to the theorems are quite straight forward and can be found in the extended version of the paper [20].

Theorem 1 (Token Unforgeability). *Given a Service Provider's public key, and a poly-bounded (in a security parameter k) number of signatures from the Service Provider on arbitrary messages, a user is not able to generate one more token (i.e. signature on the hash of an address) except with negligible probability $\epsilon(k)$.*

Remark 1. This implies that badmouthing attacks are not possible on this system. Ballot-stuffing attacks, however, cannot be completely mitigated, as a service provider can freely issue tokens. The currency introduced in this protocol helps reducing the risk of ballot stuffing, as the service provider is limited on the number of tokens he can issue, by the number of coins he owns.

Theorem 2 (Reputation Unforgeability). *Given the public blockchain history, no service provider is able to advertise a reputation that is not its own (except with negligible probability $\epsilon(k)$).*

Theorem 3 (Customer Anonymity). *Given a rating published in the blockchain concerning a given service provider, the identity from the customer that originated this rating is indistinguishable, from the service provider's point of view, among all the customers that were previously involved in a transaction with that service provider.*

Theorem 4 (Customer Report Unlinkability). *Given two different reports, it is not possible to determine whether they were issued by the same customer or not better than guessing at random.*

The proposed protocol is able to hide the identity of the rater among all the customers who interacted with a given service provider in a certain time interval, therefore providing indistinguishability. We can devise a simple model that will give an idea of the indistinguishability of the customer reviews. Suppose that

customers purchase goods from a given service provider. The arrival of customers can be modeled by a Poisson process with parameter λ. We can also suppose that, after receiving their good, customers will wait for a certain amount of time before submitting a review. In our model, the customer will wait for a time T that is uniformly distributed over $[0; \tau]$ for some $\tau > 0$. In this case, we have the following property:

Theorem 5 (Indistinguishability). *If the arrival of customers is modeled as a Poisson process of parameter λ and if customers wait for a duration that is uniformly distributed over $[0; \tau]$ before submitting their review, then the identity of a customer will be indistinguishable over a set of $\lambda\tau$ customers in average.*

6.2 Robustness Against Generic Attacks

In this section, we will explain how our proposed system copes with generic attacks against reputation systems: bad-mouthing, ballot stuffing, Sybil attacks, and whitewashing.

Bad-mouthing. Bad-mouthing consists in lying about the performance of a service provider in order to decrease his reputation. This could be done, for example, by a competitor. Our system prevents bad-mouthing thanks to the usage of **tokens**, and this attack is prevented because token unforgeability is guaranteed by the system.

Ballot-stuffing. Ballot stuffing is the opposite of bad-mouthing. This attack consists in increasing one's own reputation. As the service providers generate the tokens that allow feedback-submission on their own, this attack could only partially be mitigated with the use of **coins**, as explained in the remark concerning token unforgeability.

Whitewashing. Whitewashing consists in exiting a system after having accumulated bad reputation, in order to re-enter it again and removing the accumulated bad reputation. As the initial reputation of a new service provider is 0, the service provider would not gain much from leaving and re-entering the system with a new identity. However, bad reviews are worse than no reviews, so there could be an incentive in order to do so. One way to limit this would be to bind the identity of a service provider to, for example, his website, through a specific operation on the blockchain. The service provider could still change the domain name, but again, this would cost money.

Sybil Attacks. Sybil attacks combine more or less the attacks described above. They consist in creating multiple identities in the system in order to disrupt it. They pose no more threat than the other types of attacks, as there is no concept of "identity" for the customers in our system, and creating multiple identities for a service provider can only be used to either perform whitewashing (if he creates one identity after another) or ballot-stuffing (if he creates multiple fake transactions).

7 Conclusion

Reputation systems need to be privacy-preserving in order to work properly, without the raters having to be afraid of retaliation. Building a reputation system that is privacy-preserving without any trust assumptions is not a trivial task. However, such a system would be highly valuable, because there is much less risk that the privacy of the users could be breached. We described such a reputation system for e-commerce applications, and analyzed the security guarantees. Some points would still need attention in future work, such as the exact way of generating coins that would ensure that service providers have enough of them in order to be able to supply enough tokens for their customers, but at the same time still limit ballot-stuffing attacks. Also, we must find a definite way to address the problem of information leakage concerning the time at which the reviews are submitted.

References

1. Resnick, P., Zeckhauser, R., Swanson, J., Lockwood, K.: The value of reputation on ebay: a controlled experiment. Exp. Econ. **9**(2), 79–101
2. Resnick, P., Zeckhauser, R.: Trust among strangers in internet transactions: empirical analysis of ebays reputation system. Econ. Internet E-commer. **11**(2), 23–25 (2002)
3. Anceaume, E., Guette, G., Mazenc, P.L., Prigent, N., Tong, V.V.T.: A Privacy Preserving Distributed Reputation Mechanism, October 2012
4. Lajoie-Mazenc, P., Anceaume, E., Guette, G., Sirvent, T., Tong, V.V.T.: Efficient Distributed Privacy-Preserving Reputation Mechanism Handling Non-Monotonic Ratings, January 2015
5. Bethencourt, J., Shi, E., Song, D.: Signatures of reputation. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 400–407. Springer, Heidelberg (2010)
6. Pavlov, E., Rosenschein, J.S., Topol, Z.: Supporting privacy in decentralized additive reputation systems. In: Jensen, C., Poslad, S., Dimitrakos, T. (eds.) iTrust 2004. LNCS, vol. 2995, pp. 108–119. Springer, Heidelberg (2004)
7. Dimitriou, T., Michalas, A.: Multi-party trust computation in decentralized environments in the presence of malicious adversaries. Ad Hoc Netw. **15**, 53–66 (2014)
8. Hasan, O., Brunie, L., Bertino, E., Shang, N.: A decentralized privacy preserving reputation protocol for the malicious adversarial model. IEEE Trans. Inf. Forensics Secur. **8**(6), 949–962 (2013)
9. Kerschbaum, F.: A verifiable, centralized, coercion-free reputation system. In: Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society, WPES 2009, pp. 61–70. ACM, New York (2009)
10. Bitcoin, S.N.: A peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
11. Pilkington, M.: Blockchain technology: principles and applications. In: Olleros, F.X., Zhegu, M. (eds.) Research Handbook on Digital Transformations. Edward Elgar, Northampton (2016)
12. Carboni, D.: Feedback based reputation on top of the bitcoin blockchain. arXiv preprint arXiv:1502.01504 (2015)

13. Androulaki, E., Choi, S.G., Bellovin, S.M., Malkin, T.: Reputation systems for anonymous networks. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 202–218. Springer, Heidelberg (2008)
14. Petrlic, R., Lutters, S., Sorge, C.: Privacy-preserving reputation management. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC 2014, pp. 1712–1718. ACM, New York (2014)
15. Comodo report of incident - comodo detected and thwarted an intrusion on 26 march 2011. https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html
16. Key internet operator verisign hit by hackers. http://www.reuters.com/article/2012/02/02/us-hacking-verisign-idUSTRE8110Z820120202
17. Maintaining digital certificate security. http://googleonlinesecurity.blogspot.fr/2015/03/maintaining-digital-certificate-security.html
18. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy, SP 2008, pp. 111–125, May 2008
19. Barbaro, M., Zeller, Jr., T.: A face is exposed for aol searcher no. 4417749, August 2006
20. Schaub, A., Bazin, R., Hasan, O., Brunie, L.: A trustless privacy-preserving reputation system. Cryptology ePrint Archive, Report /016 (2016). http://eprint.iacr.org/
21. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
22. Okamoto, T., Ohta, K., Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
23. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (2001)
24. Jøsang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Electronic Commerce Conference, vol. 5, pp. 2502–2511 (2002)

Author Index