IFIP AICT 354

Jan Camenisch
Simone Fischer-Hübner
Yuko Murayama
Armand Portmann
Carlos Rieder
(Eds.)

Future Challenges in Security and Privacy for Academia and Industry

26th IFIP TC 11 International
Information Security Conference, SEC 2011
Lucerne, Switzerland, June 2011
Proceedings

Springer

IFIP Advances in Information and Communication Technology 354

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. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

> *IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.*

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- Open conferences;
- Working conferences.

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 small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is less 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.

Any national society whose primary activity is in information may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly, National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

Jan Camenisch Simone Fischer-Hübner
Yuko Murayama Armand Portmann
Carlos Rieder (Eds.)

Future Challenges in Security and Privacy for Academia and Industry

26th IFIP TC 11 International
Information Security Conference, SEC 2011
Lucerne, Switzerland, June 7-9, 2011
Proceedings

Springer

Volume Editors

Jan Camenisch
IBM Zurich Research Laboratory
Säumerstr. 4, 8803 Rüschlikon, Switzerland
E-mail: jca@zurich.ibm.com

Simone Fischer-Hübner
Karlstad University, Department of Computer Science
Universitetsgatan 1, 65188 Karlstad, Sweden
E-mail: simone.fischer-huebner@kau.se

Yuko Murayama
Iwate Prefectural University, Faculty of Software and Information Science
152-52 Sugo, Takizawa, Takizawa-mura, Iwate 020-0193, Japan
E-mail: murayama@iwate-pu.ac.jp

Armand Portmann
Carlos Rieder
Lucerne University of Applied Sciences and Arts
Zentralstr. 9, 6002 Lucerne, Switzerland
E-mail: {armand.portmann, carlos.rieder}@hslu.ch

ISSN 1868-4238 e-ISSN 1868-422X
ISBN 978-3-642-21423-3 e-ISBN 978-3-642-21424-0
DOI 10.1007/978-3-642-21424-0
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011927858

CR Subject Classification (1998): C.2, K.6.5, D.4.6, E.3, H.4, J.1

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book contains the proceedings of the 26th IFIP TC-11 International Information Security Conference (IFIP/SEC 2011) on "Future Challenges in Security and Privacy for Academia and Industry" held during June 7–9, 2011, at the Lucerne University of Applied Sciences and Arts, Switzerland.

The SEC conferences are in a series of well-established international conferences on security and privacy organized annually by the Technical Committee 11 (TC-11) of IFIP (International Federation for Information Processing). IFIP SEC 2011 aimed at bringing together primarily researchers, but also practitioners from academia, industry and governmental institutions to elaborate and discuss the IT security and privacy challenges that we face today and in the future. Papers offering novel and mature research contributions, on any aspect of information security and privacy, were solicited for submission to the 26th IFIP TC-11 International Information Security Conference.

IFIP SEC 2011 received 100 submissions which were all reviewed by at least three members of the international Program Committee (PC). Based on an intensive discussion among the reviewers and other PC members, 24 papers were selected for presentation at the conference. Topics addressed by the accepted papers published in these proceedings include authentication, intrusion detection, malware, information flow and DoS attacks, network security and security protocols, policy compliance and obligations, privacy attacks and privacy-enhancing technologies, risk analysis and security metrics as well software security.

Further highlights of IFIP SEC 2011 were the three invited keynote presentations by high-ranked IT security and privacy experts: The recipient of the 2011 Kristian Beckman award granted by IFIP TC-11—Ann Cavoukian, Information and Privacy Commissioner (IPC) of Ontario, Canada, as well as Michael Waidner, Director of Fraunhofer SIT Darmstadt, Germany, and René Hüsler from the Lucerne University of Applied Sciences and Arts, Switzerland. The paper for the invited keynote by Ann Cavoukian is also included in these proceedings.

In addition to the invited keynote and accepted paper sessions, IFIP SEC 2011 also included an industrial track on "Research Meets Industry" as well as the following four workshops and sub-conferences: The workshops organized by the EU FP7 projects PrimeLife and PICOS, the iNetSec 2011 organized by IFIP Working Group 11.4, as well as the 7th World Conference on Information Security Education WISE7 organized by IFIP Working Group 11.8. WISE7 and iNetSec 2011 were organized autonomously by the respective IFIP Working Groups. They had their own Call for Papers and Program Committees and the accepted papers are published in their own proceedings.

IFIP SEC 2011 was organized by Lucerne University of Applied Sciences and Arts. We would like to thank UBS AG, Zurich/Switzerland, Crypto AG, Zug/Switzerland, Elsevier Limited, Oxford/UK and isec ag, Lucerne/Switzerland, for

sponsoring IFIP SEC 2011. Besides, we gratefully acknowledge all authors, members of the Program Committee and additional reviewers for their contributions to the scientific quality of this conference. Last but not least, we owe thanks to the Organizing Committee, and especially to its Chair Carlos Rieder, for all the efforts and dedication in preparing this conference.

June 2011

Jan Camenisch
Simone Fischer-Hübner
Yuko Murayama
Armand Portmann

Organization

Program Committee Chairs

Jan Camenisch	IBM Research - Zurich, Switzerland
Simone Fischer-Hübner	Karlstad University, Sweden
Yuko Murayama	Iwate Prefectural University, Japan

Publication Chair

Armand Portmann	Lucerne University of Applied Sciences and Arts, Switzerland

Program Committee Members

Ejaz Ahmed	Queensland University of Technology, Australia
Colin Armstrong	Gailaad Pty. Ltd., Australia
Vijay Atluri	Rutgers University, USA
Richard Baskerville	Georgia State University, USA
Bharat Bhargava	Purdue University, USA
Katrin Borcea-Pfitzmann	T.U. Dresden, Germany
Reinhardt Botha	NMMU, South Africa
David Chadwick	University of Kent, UK
Nathan Clarke	University of Plymouth, UK
Roger Clarke	Xamax Consultancy Pty. Ltd., ANU and UNSW, Australia
Nora Cuppens-Boulahia	TELECOM Bretagne, France
Ed Dawson	QUT, Australia
Sabrina de Capitani di Vimercati	Università degli Studi di Milano, Italy
Bart de Decker	K.U. Leuven, Belgium
Yves Deswarte	LAAS-CNRS, France
Ronald Dodge	U.S. Military Academy, USA
Jan Eloff	University of Pretoria, South Africa
Sarah Foresti	Università degli Studi di Milano, Italy
Felix Freiling	Mannheim University, Germany
Lothar Fritsch	Norwegian Computer Center, Norway
Steven Furnell	University of Plymouth, UK
Mark Gasson	University of Reading, UK

Dieter Gollmann	TU Hamburg-Harburg, Germany
Dimitris Gritzalis	Athens University of Economics and Business, Greece
Stefanos Gritzalis	University of the Aegean, Greece
Marit Hansen	Independent Center for Privacy Protection, Germany
Alejandro Hevia	University of Chile, Chile
Jaap-Henk Hoepmann	University of Twente, The Netherlands
René Hüsler	Lucerne Univiversity of Applied Sciences and Arts, Switzerland
Cynthia Irvine	Naval Postgraduate School, Monterey, USA
Sushil Jajodia	George Mason University, USA
David-Olivier Jaquet-Chiffelle	Bern University of Applied Sciences, Switzerland
Lech Janczweski	University of Auckland, New Zealand
Dogan Kesdogan	University of Siegen, Germany
Valentin Kisimov	University of World and National Economy Sofia, Bulgaria
Stefan Köpsell	T.U. Dresden, Germany
Stewart Kowalski	DSV/Stockholm University (and Huawei), Sweden
Ioannis Krontiris	Goethe University Frankfurt, Germany
Lam-For Kwok	City University of Hong Kong, Hong Kong
Costas Lambrinoudakis	University of the Aegean, Greece
Carl E. Landwehr	University of Maryland, USA
Ronald Leenes	Tilburg University, The Netherlands
Herbert Leitold	Technical University of Graz, Austria
Stefan Lindskog	Karlstad University, Sweden
Javier López	Universidad de Malaga, Spain
Luigi Lo Iacono	EUFH Bruehl, Germany
Steve Marsh	Communications Research Center Canada, Canada
Fabio Martinelli	National Research Council, Italy
Leonardo Martucci	CASED, Germany
Václav Matyás	Masaryk University, Brno, Czech Republic
Carlos Maziero	University of Parana, Brazil
Natalia Miloslavskaya	MEPHI, Russia
Refik Molva	Institut Eurecom, France
Eiji Okamoto	University of Tsukuba, Japan
Rolf Oppliger	eSecurity, Switzerland
Jakob-Illeborg Pagter	Alexandra Instituttet AS, Denmark

George Pangalos	University of Thessaloniki, Greece
Jong-Hyuk Park	Kyungnam University, South Korea
Philippos Peleties	Universal Bank Ltd., Cyprus
Günther Pernul	University of Regensburg, Germany
Ulrich Pinsdorf	Microsoft EMIC, Germany
Hartmut Pohl	University of Applied Sciences Bonn-Rhein-Sieg, Germany
Roland Portmann	Lucerne University of Applied Sciences and Arts, Switzerland
Kai Rannenberg	Goethe University Frankfurt, Germany
Marc Rennhard	Zurich University of Applied Sciences, Switzerland
Carlos Rieder	Lucerne University of Applied Sciences and Arts, Switzerland
Rodrigo Roman	University of Malaga, Spain
Andrei Sabelfeld	Chalmers University of Technology, Sweden
Pierangela Samarati	University of Milan, Italy
Ingrid Schaumüller-Bichl	Upper Austria University of Applied Sciences, Austria
Anne Karen Seip	Financial Supervisory Authority of Norway, Norway
Nahid Shahmehri	Linköping University, Sweden
Siraj Shaikh	Coventry University, UK
Einar Snekkenes	Gjøvik University College, Norway
Miquel Soriano	UPC, Spain
Sandra Steinbrecher	Technical University of Dresden, Germany
Rama Subramaniam	Valiant Technologies, India
Willy Susilo	University of Wollongong, Australia
Stephanie Teufel	University of Freiburg, Switzerland
Bill Tsoumas	Ernst & Young, Greece
Pedro-Manuel Veiga	Universidade de Lisboa, Portugal
Hein Venter	University of Pretoria, South Africa
Teemupekka Virtanen	Helsinki University of Technology, Finland
Melanie Volkamer	CASED, Germany
Rossouw von Solms	Nelson Mandela Metropolitan University, South Africa
Jozef Vyskoc	VaF, Slavak Republic
Christian Weber	Goethe University Frankfurt, Germany
Tatjana Welzer	University of Maribor, Slovenia
Rigo Wenning	W3C, France
Sven Wohlgemuth	National Institute of Informatics, Japan

Louise Yngström	University of Stockholm, Sweden
Jianying Zhou	I2R, Singapore

Additional Reviewers

Gergely Alpár
Gökhan Bal
Mohamed Bourimi
Christian Broser
Laurent Bussard
Sebastian Clauß
Denise Demirel
Anthony Dessiatnikoff
Andreas Dewald
Stelios Dritsas
Thomas Fielenbach
Christoph Fritsch
Viiveke Fåk
Dimitris Geneiatakis
Mariana Gerber
Shkodran Gerguri
Stephan Groß
Daniel Hedin
Stephan Heim
Christos Ilioudis
Maarten Jacobs
Thomas Jakobsen
Fatih Karatas
Jonathan Katz
Mohamed Kaâniche
Benjamin Kellermann
Nizar Kheir
Marc-Olivier Killijian
Leanid Krautsevich
Harsha Kumara
Jorn Lapon
Anja Lehmann
Dimitrios Lekkas
Jonas Magazinius
Ilaria Matteucci
Nasir Memon
Vincent Naessens
Michael Niedermeier
Janus Dam Nielsen
Alexandros Papanikolaou
Vinh Pham
Franz-Stefan Preiss
Klaus Rechert
Andreas Reisser
Moritz Riesner
Panagiotis Rizomiliotis
Jan Schlüter
Andriy Stetsko
Tim Storer
Petr Svenda
Marianthi Theoharidou
Aggeliki Tsohou
Pavel Tucek
Simeon Veloudis
Nikolaos Virvilis
Stefan Voemel
Carsten Willems
Lars Wolos
Hau-San Raymond Wong
Erik Wästlund
Thomas Zefferer
Bernd Zwattendorfer

Table of Contents

Software Security

Policy Compliance and Obligations

Privacy Attacks and Privacy-Enhancing Technologies

Risk Analysis and Security Metrics

Intrusion Detection

Appendix

Patience, Persistence, and Faith: Evolving the Gold Standard in Privacy and Data Protection

Ann Cavoukian

Information and Privacy Commissioner, Ontario

Abstract. *Privacy by Design* (*PbD*) is a concept that was developed by Ontario's Information and Privacy Commissioner, Dr. Ann Cavoukian, in the '90s. It prescribes that privacy be embedded directly into the design and operation, not only of various technologies, but also of business processes and networked infrastructure. Instead of treating privacy as an after-thought – "bolting it on after the fact" – *PbD* is proactive and preventative in nature.

Through years of advocacy and encouragement, PbD is now being widely adopted globally by a growing number of organizations and jurisdictions. This paper outlines the foundations of PbD, and traces its evolution from a conceptual framework into a practical one that has been recognized internationally as the gold standard in privacy and data protection.

Personal information, be it biographical, biological, genealogical, historical, transactional, locational, relational, computational, vocational, or reputational, is the substance that makes up our modern identity. Our digital footprints and shadows are being gathered together, bit by bit, megabyte by megabyte, terabyte by terabyte, into personas and profiles and avatars – virtual representations of ourselves that exist in thousands of simultaneous locations. These technologies give us access to extraordinary new services, conveniences, efficiencies and benefits, undreamt of by our parents. At the same time, novel risks and unimagined threats are emerging from this digital cornucopia. Identity fraud and theft are the diseases of the Information Age, along with new forms of deception and social engineering made possible by the surfeit of data.

These developments have prompted some critics to pronounce that privacy is either dead or dying[1]. I don't believe that to be the case, but there is no question that our fundamental ideas about identity and privacy, the strategies that we have collectively pursued, and the technologies that we have adopted, must evolve and adapt to keep apace with our rapidly changing world of connectivity, networking, participation, sharing, and collaboration.

[1] See, for example, SimsonGarfinkel, Database Nation: The Death of Privacy in the 21st Century. O'Reilly Media, California (2000); Robert O'Harrow, No Place to Hide: Behind the Scenes of our Emerging Surveillance Society. Free Press, New York (2005); David Brin, The Transparent Society. Addison-Wesley, New York (1998); and Jerry Rosenberg, The Death of Privacy.Random House, New York (1969).

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 1–16, 2011.

At stake is not only our privacy, but also the consumer confidence and trust that underpins and enables today's information society. What will privacy mean, and how will privacy survive, and hopefully thrive, as a viable human right, operational value, and critical enabling trust factor, in a world where the individual is increasingly removed from personal involvement in data-rich transactions? What will informational self-determination – the basis of current privacy laws and practices – mean when data is increasingly stored and processed away from personal computing devices, in the world of a nebulous "Cloud?"

These are precisely the kinds of questions I have been grappling with in my 20 year career in privacy. My attempts to answer these questions – to imagine a future where privacy continues to exist in some form we find recognizable – are the foundations of the work for which I am honoured to have received the Kristian Beckman Award. *Privacy by Design* – the concept I pioneered relating to engineering privacy directly into the design of new technologies, accountable business practices, and networked infrastructure as a core functionality – is an approach I have advocated for many years. It has, only recently, reached a tipping point, and now appears to be gathering momentum around the world.

This paper traces the roots of the concept, and chronicles some of the highlights in my ongoing crusade to see *Privacy by Design* implemented with sufficient breadth and depth that it effectives assures a future for privacy and all the social values it enables.

Privacy Will Always Matter

Privacy is not dead, and will never die, given the essential role it plays in preserving our freedoms, but it is, perhaps, beleaguered. Practical obscurity – the basis for privacy norms throughout much of history – is fast disappearing. The functional impediments to surveillance that once protected privacy, by default – such as data processing and storage costs, and the difficulty of linking files from multiple databases – are increasingly irrelevant.

Privacy, famously described by American Justices Samuel Warren and Louis Brandeis as "the right to be let alone,"[2] is closely related to individual dignity and integrity, personal autonomy, freedom of association, and independence. It is the underpinning of many of the rights and values we hold dear, and has long been – and still remains – a vital component of free and democratic societies. Historically, when a society devolves from a free and democratic one into a totalitarian state, privacy is the first thread to unravel.

Our need to preserve private spaces in our lives, to permit intimacy, and to enjoy solitude, is as relevant now as it has ever been. Indeed, it is perhaps *more* relevant now that our lives are so networked, inter-connected, and "plugged in."

The idea that privacy is under attack is not new, but the weapons of choice have changed over time. Over hundreds of years, critics have raised concerns about the privacy implications of almost every new technology, from the camera to the telephone to the personal computer. And those who have taken it upon themselves to protect our privacy have been forced to innovate similarly, developing new strategies and expanding their arsenals as new threats emerge.

[2] Warren, S. and Brandeis, L.: *The Right to Privacy*. Harvard Law Review 4, 193-220 (1890).

One Giant Leap: The Story of PETs

If privacy is gasping for air, it is most certainly, at least in part, because traditional ways of preserving privacy are no longer sufficient or relevant. This has been the case for quite some time.

But in the 1990's, the seeds of a way forward were planted when it began to become clear that our dated approach to protecting privacy, which was based on the assumption that privacy and technology were necessarily opposed to one other, had no viable future. The forward march of technology could not – and arguably should not – be stopped. Somehow, privacy had to find a way to live on.

This led to the development of Privacy-Enhancing Technologies (PETs), which are predicated on the idea of enlisting the support of technology to *enhance* privacy, rather than *encroach* upon it. Believe it or not, the idea was a radical one at the time.

PETs are information and communications technologies that strengthen the protection of personal privacy in an information system by preventing the unnecessary or unlawful collection, use, and disclosure of personal data, or by offering tools to enhance an individual's control over his/her personal data.

The concept grew, in part, out of feeling that encryption technologies could help individuals and organizations protect personal information in the face of the widespread dissemination of personal computers and the advent of the Internet as a (then) new medium of communications. Western governments, however, were trying to restrict the use and export of encryption products, and engineer surveillance "back-doors" into the emerging digital telecommunications infrastructures. This met with fierce resistance from cryptographers, privacy advocates, rights groups, and business interests. If new information and communications technologies threatened to invade individual privacy, the thinking went, then these types of privacy-enhancing technologies, that could empower individuals and restore trust, were the solution.

We first advanced the concept of "PETs" in a collaborative work between my office and the Netherlands Data Protection Authority in 1995.[3] From the outset, PETs emphasized the need to incorporate the universal principles of Fair Information Practices (FIPs) – universal privacy principles for handling personal data –into the actual code and operation of information processing technologies and systems.

First codified by the OECD in 1980, there are many articulations of Fair Information Practices, including the E.U. Directive on Data Protection, Canada's CSA Privacy Code, the Asia-Pacific Economic Cooperation (APEC) Privacy Framework, the U.S. Safe Harbor Principles, and the harmonized Global Privacy Standard.[4] Despite minor differences in language and emphasis, these FIPs all reflect the following fundamental concepts:

---

[3] Information and Privacy Commissioner/Ontario, Registratiekamer/TheNetherlands: Privacy-Enhancing Technologies: The Path to Anonymity (Volume I). (1995) `http://www.ipc.on.ca/English/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=329`

[4] Information and Privacy Commissioner/Ontario, Creation of a Global Privacy Standard (2006): `www.ipc.on.ca/images/Resources/gps.pdf`

- **Purpose Specification and Use Limitation** – reasons for the collection, use, disclosure and retention of personally identifiable information should be identified at or before the time of collection. Personal information should not be used or disclosed for purposes other than those for which it was collected, except with the consent of the individual or as required by law;
- **User participation** – individuals should be empowered to play a participatory role and exercise controls during the life cycle of their own personal data;
- **Strong security** – the confidentiality, integrity and availability of personal data should be safeguarded, as appropriate to the sensitivity of the information.

In their design and implementation, PETs should ideally promote *all* of these meta-principles. The use of strong encryption technologies to secure detailed customer records against unauthorized access, for example, is extremely valuable, but it speaks little to data minimization and user participation. Building in privacy principles early and comprehensively into information technologies and systems is central to good PETs and would later, through a process of incremental refinement, become the anchor for my trademark *Privacy by Design*approach.

Traditional PETs contribute to the achievement of the privacy ideal of informational self-determination – the *individual's* ability to exercise a measure of control over the collection, use and disclosure of their personal information. They have typically been defined as performing the following functions:

- preventing unauthorized access to personal communications and stored files;
- automating the retrieval of information about data collectors' privacy practices and automating users' decision-making on the basis of these practices;
- preventing automated data capture through cookies, HTTP headers, web bugs, spyware, etc.;
- preventing communications from being linked to a specific individual;
- facilitating transactions that reveal minimal personal information; and
- filtering unwanted messages.

As PETs are user-centric tools and functions, this list has not been significantly lengthened in over a decade. Over time, we began to wonder whether we had perhaps placed unnecessary boundaries around PETs. Were they cryptographic primitives, software or hardware applications, components embedded in larger systems, or entire information systems? Should PETs be understood to include only technologies under the exclusive control of the individual, or was there room for a more expansive definition that included important and complementary infrastructure components beyond the control of the individual?

PETs Plus: Putting a Positive Spin on Privacy

Within 10 years of the concept of PETs being clearly articulated, the "new normal" of surveillance – deeper and broader than ever before – was such that the limitations of PETs were becoming clear. The door to a more expansive understanding had to be opened.

PETs embody fundamental privacy principles by minimizing personal data use, maximizing data security, and empowering individuals. They are useful, but no longer sufficient in and of themselves to assure sustained, meaningful privacy protection. And while FIPs remained relevant, a significant challenge was that the early drafters and adopters of FIPs clearly had in mind large mainframe computers and centralized electronic databases. They could never have imagined how leapfrogging revolutions in sensors, bandwidth, storage, and processing power would converge into our current hyper-connected Web 2.0 world of ubiquitous data availability.

A second challenge with the PETs approach was that it kept the privacy conversation contained to a relatively small suite of technologies, and therefore marginalized. Further, organizations were inclined to think that they could have technology systems that *either* protected privacy *or* were effective in meeting their business objectives – not both. The kinds of arguments that were being made in support of PETs did little to disabuse them of this notion.

As surveillance technologies continued to expand throughout the 1990's, I observed that privacy was constantly losing out in debates that pitted it against values like public safety, security, and even efficiency. Some of the surveillance control technologies that privacy was losing out to at the time included (typical objectives shown in brackets):

- Public and private video surveillance (public safety)
- Employee monitoring and surveillance (corporate data security)
- Network monitoring, profiling and database analytics (network forensics, marketing)
- Device location tracking (safety, resource allocation, marketing)
- "Whole of customer" transaction aggregation (customer service)
- Creation and uses of "enriched" profiles to identify, verify and evaluate (security)
- Creation and uses of interoperable biometric databases (access control/security)

These types of surveillance systems were being built around the basic assumption that users/subjects had to give up some of their privacy in order to benefit from improved system security and functionalities. This is how privacy was being increasingly "trumped" by social, legal, and economic imperatives: it was being characterized as a zero-sum tradeoff – always coming at the expense of other interests against which it had to be "balanced."

But balance metaphors assume that the two interests being balanced are always in conflict, and that an increase in one necessarily translates into a decrease in the other. More privacy equals less security; more security equals less privacy.[5] This simply isn't the case.[6]

[5] See Julian Sanchez's excellent blog posting (February 4, 2011) on the shortcomings of balance metaphors at www.juliansanchez.com, which is based on Orin Kerr's An Equilibrium-Adjustment Theory of the Fourth Amendment. Harvard Law Review, Vol. 125 (forthcoming) http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1748222.

[6] Indeed, the balance metaphor is coming under growing criticism. See, for example, Daniel Solove: Nothing to Hide: The False Tradeoff Between Privacy and Security. Yale University Press (forthcoming, 2011).

Ultimately, PETs were effective in some situations, but less so in others, and their long-term strategic value to advancing the cause of privacy into the mainstream appeared to be limited. In my view, PETs *Plus* – Privacy-Enhancing Technologies applied in the context of a positive-sum, not zero-sum paradigm – represented the next evolution of PETs.

By adding a "positive-sum" outlook to the design and use of information and communication technologies, PETs *Plus* made it possible to conceive of achieving goals *beyond* privacy, while *also* achieving privacy goals. It recognized the legitimate goals of other participants or stakeholders in the development process, such as, for example, those of the system owner and operator, in a positive-sum rather than zero-sum, "either/or" model. Thus the functional and operational objectives of a system (e.g., to transport and route electronic communications, to process a payment, or provide a service), and other security, surveillance, and anti-fraud detection goals, could be achieved while *also* protecting privacy. It was a conceptual shift that would, over time, prove critical in engaging organizations meaningfully in the privacy issue. The time had come to move beyond false dichotomies and short-sighted "balancing acts."

Beyond Individual Responsibility

Throughout this period, changes in information technology were making it increasingly difficult for individuals to exert meaningfulcontrol over the collection, use, and disclosure of their personal information. Significantly, at about the same time, organizations were beginning to face pressure to provide better privacy assurances, for a number of reasons.[7]

By the mid-'90s, there was considerable public discussion in the European Union, Canada and the United States about the merits of good privacy practices flowing from the anticipated coming into force of the European Data Protection Directive.[8]The EU Directive sought to strike a balance between a high level of protection for the privacy of individuals and the free movement of personal data. Significantly, when transposed to EU Member national law, the EU Directive would require foreign jurisdictions and businesses to meet its "adequacy" requirements in order to receive transfers of any personal information about EU citizens.

In Canada, a broad coalition of business and consumer interests was meeting to establish a national, voluntary privacy code to guide the legitimate information requirements of business, industry and institutions operating in the information age. Their efforts would ultimately be legislated in the *Personal Information Protection and Electronic Documents Act* (PIPEDA). In the United States, negotiations began with the EU on a "SafeHarbor" framework agreement to establish similar ground rules for the processing of personal information by U.S. businesses.

---

[7] In 1995, Don Tapscott and I co-authored Who Knows: Safeguarding your Privacy in a Networked World. Random House, Toronto (1995) that captures the spirit of this time.

[8] European Parliament and Council Directive 95/46/EC of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.Official Journal L 281 of 23.11. (1995).

Aside from these regulatory developments, growing interest in electronic commerce was also putting a new spotlight on privacy. The fulfillment of the promise of the Information Age would rely, in large measure, on the ability to foster the confidence and trust necessary for active consumer participation. Against this background, my office published several white papers such as,*Privacy Protection Makes Good Business Sense,*[9] and *Privacy: The Key to Electronic Commerce*[10]that argued that any organization that collects, uses and/or discloses personal information should proactively accommodate the privacy interests and rights of individuals, throughout its operations. More than a moral imperative, respecting privacy would offer a business "payoff" to organizations in the form of: improved customer satisfaction and trust; enhanced reputations; reduced legal liabilities; more efficient operations; commercial gains and enhanced ROI; and, ultimately, an enduring competitive advantage.

Toward a New Paradigm: *Privacy by Design*

Against this backdrop, the Ontario Government, like many other public and private sector organizations, had, since the mid 1990's, begun to adopt increasingly sophisticated information and communications technologies in an effort to leverage the benefitsof the emerging "Information Highway," as it was then called. Of course, the collection, use, sharing and retention of more and more personal information, made possible by these large-scale IT projects, posed significant privacy issues.

Given my office's role in overseeing provincial and municipal government compliance with access to information and privacy laws, and my position on privacy and technology issues, I was increasingly being consulted by the government, as well as other public and private sector organizations, for advice and guidance on how, exactly, to proceed. The answer involved being proactive and building privacy in*early on* — at the design stage of these new systems, namely, *Privacy by Design*.

What followed was a succession of joint collaborations[11] on groundbreaking new technology-enabled projects that focused on developing and applying Privacy by Design principles into the development process so that any privacy-invasive risks could either be minimized or eliminated altogether.

---

[9] Information and Privacy Commissioner/Ontario, Privacy Protection Makes Good Business Sense, www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=327

[10] Information and Privacy Commissioner/Ontario, Privacy: The Key to Electronic Commerce, www.ipc.on.ca/images/Resources/e-comm.pdf

[11] See, for example, Information and Privacy Commissioner/Ontario publications: Smart, Optical and Other Advanced Cards: How to do a Privacy Assessment, www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=297; 407 Express Toll Route: How You Can Travel the 407 Anonymously,www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=335; Intelligent Software Agents: Turning a Privacy Threat into a Privacy Protector, www.ipc.on.ca/images/Resources/up-isat.pdf; Privacy Design Principles for an Integrated Justice System - Working Paper, www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=318

All of these elements came together that year in my presentation, *Privacy by Design: Building Trust into Technology* to the 1st Annual Privacy and Security Workshop by the Centre for Applied Cryptographic Research (CACR) in 2000.[12]

At the same time, market changes, technological developments, and evolutions within the privacy community converged such that by 2000, when the deeply influential Computers, Freedom and Privacy (CFP) conference held its 10th annual meeting in Toronto, organizers and participants set aside – for the first time – their traditional focus on legislative privacy protections.

In an exploratory *Workshop on Freedom and Privacy by Design*, participants considered how technology could be leveraged to bring about strong protections of civil liberties that would be guaranteed by the technologies themselves.[13] The workshop aimed at developing principles for designing and implementing information architectures, strategies and evaluation criteria that could be inherently privacy protective. To do so, it brought together programmers, cryptographers, and systems architects with lawyers, social scientists, writers and user/experts.

Still, their focus remained largely technological. By contrast, my office's work on *Privacy by Design*, though rooted in PETs, recognized the need to embed privacy at the design stages of information technologies, architectures, *and* systems. By applying 7 foundational principles, the objectives of *Privacy by Design* could be met in all of these areas.

The 7 Foundational Principles of *Privacy by Design*

1. Proactive not Reactive; Preventative not Remedial
2. Privacy as the Default Setting
3. Privacy Embedded into Design
4. Full Functionality – Positive-Sum, not Zero-Sum
5. End-to-End Security – Full Lifecycle Protection
6. Visibility and Transparency – Keep it Open
7. Respect for User Privacy – Keep it User-Centric

These principles are applied within the context of data minimization – the idea that the collection, use, disclosure and retention of personal information should be minimized wherever, and to the fullest extent, possible. This concept, which had been missing from most articulations of Fair Information Practices, appeared in the Global

[12] Presentation by Ann Cavoukian, Ph.D. to the 1st Annual Privacy and Security Conference, Centre for Applied Research, *Privacy by Design: Building Trust into Technology,* www.cacr.math.uwaterloo.ca/conferences/2000/isw-sixth/cavoukian.ppt

[13] Computers, Freedom and Privacy workshop proceedings, www.cfp2000.org/ workshop/materials/

Privacy Standard that was developed by a Working Group of Commissioners that I chaired, and that had as its sole focus the creation of an internationally harmonized set of FIPs.[14]

My broader, more holistic approach to protecting privacy, by design, would begin to take hold in the first decade of the new millennium. *Privacy by Design* would also begin to move the privacy debate out of the win/lose, zero-sum paradigm. As a result, it would challenge organizations to think creatively about how all system objectives – including privacy – could be met. This opened the door for actively bringing privacy into the development process – a substantial break with traditional approaches that often left privacy to the last minute or later (to the extent that it was addressed at all), placing the bulk of the responsibility in the hands of individual consumers.

Privacy by Design in the Post-9/11 World: Challenges and Opportunities

In the meantime, the events and consequences of September 11, 2001 challenged assumptions among many privacy advocates, freedom fighters and technologists that individual privacy was necessarily paramount to all other interests in society. They found it increasingly difficult to defend privacy interests in an atmosphere characterized by visceral public fear and a collective desire for security.

Almost overnight, the privacy threat model changed. Governments enacted legislation and put in place initiatives that trumped traditional information privacy legislation and individual rights, often enlisting private-sector organizations to collect and use more granular personal information than ever before.

This was the classic zero-sum paradigm writ large: the more we have of one interest (public security), the less we can have of another (individual privacy). Privacy could never win out – and arguably could not even survive – within this zero-sum framework.

Unpopularly at first, I challenged the underlying premise that privacy necessarily had to be ceded in order to gain security benefits, arguing that both could be achieved at the same time. I posited that many security technologies could be redesigned to remain effective, while minimizing or eliminating their privacy invasive features. By substituting a new premise - that privacy and security were two complementary sides of an indivisible whole (not opposites), I argued that we could design technologies that protect public safety *without* sacrificing privacy. What a concept!

My approach during this period was three-pronged:

1. Challenging the privacy community to question existing paradigm assumptions, and to raise the level of debate on security and privacy above traditional, simplistic, either/or viewpoints.
2. Challenging two distinct groups: 1) legislators, policy analysts and legal counsel that draft legislation focused on security and public safety, and 2) directors, managers, individuals, who develop Requests for Proposals (RFPs) and set the

[14] Information and Privacy Commissioner/Ontario, Creation of a Global Privacy Standard. (2006), www.ipc.on.ca/images/Resources/gps.pdf

procurement 'specs' for security technologies, to be more mindful of how privacy interests could be accommodated in their work.
3. Challenging solution providers – engineers, technologists, software designers, and other developers of technology and their industry associations - to introduce privacy concepts into the policy statements of their organizations and associations, and, more importantly, to embed privacy directly into the concept, design and implementation of their technology solutions. Namely, add privacy features into code, making it a core functionality.

Over time, this view gained credence, and privacy principles began to be introduced into otherwise security-focused frameworks.

The Privacy Payoff: Taking *Privacy by Design* into the Business Community

While the security community remained an essential focus in the aftermath of September 11, it was clear to me that, in order to ensure meaningful privacy protection well into the future, I had to continue my work to entrench privacy as a basic business practice. So, during the early 2000's, I redoubled my efforts to engage the business community.

My arguments were published in 2002 as *The Privacy Payoff*,[15] a book I co-authored with Tyler Hamilton. The premise was simple: businesses were collecting information about their customers through knowledge-based technology, hoping to better serve their customers and, in turn, increase their profits. But most were overlooking one key point – customers didn't like how they went about this.

Going beyond quick fixes, *The Privacy Payoff* offered companies concrete steps to avoid the risks of the privacy minefield and reap the advantages of a privacy-sensitive corporation. It discussed global regulations and trends, drafting and implementing a privacy policy, and more. The central message – that adopting good privacy and security practices paid back multiple dividends and was highly desirable, regardless of legal and regulatory requirements – remains to this day a core axiom of the *Privacy by Design* approach.

The business case for privacy that we outlined in *The Privacy Payoff* focused, in essence, on gaining and keeping customer trust and loyalty, which, in turn, leads to repeat and higher-value business, and avoids "churn." Good privacy leads to solid ROI (return on investment). We also outlined how the privacy payoff could work in reverse: poor privacy practices could result in additional costs and foregone opportunities and revenues, along with a host of other negative consequences. These could include:

- harm to clients or customers whose personal data was used or disclosed inappropriately;
- costly damage to an organization's reputation and brand;

[15] Ann Cavoukian & Tyler Hamilton, The Privacy Payoff: How Successful Businesses Build Customer Trust. McGraw-Hill Ryerson (2002).

- financial losses associated with deterioration in the quality or integrity of personal data;
- financial losses due to a loss of business or delay in the implementation of a new product or service, due to privacy concerns;
- loss of market share or a drop in stock prices following negative publicity;
- violations of privacy laws and regulations;
- diminished confidence and trust in the industry.

The Privacy Payoff was followed by a joint publication between my office and Deloitte &Touche[16] that clarified the central issues and challenges for organizations, such as the critical distinctions and interplay between information security and privacy, and provided advice for developing strategies to enhance information security and privacy protections.

A year later, my office commissioned a study by the Ponemon Institute on corporate privacy practices.[17] The report compared what Canadian and U.S. companies were doing to achieve privacy programs and also looked at what companies were doing to move beyond simple compliance with regulations in order to build trusted relationships with stakeholders, increase revenue, and strengthen reputation and brand.

The study showed that leading companies were more likely to execute the following business practices as an integral part of their enterprise privacy program:

- Integrate information security and privacy into one virtual team
- Incorporate perspectives of legal, marketing, human resources and IT into privacy strategy
- Centralize privacy program responsibility under one senior executive sponsor
- Whenever feasible, consider using privacy enabling technologies,
- Empower local managers to get involved, especially in communications, training and outreach,
- Obtain real budget authority to implement enterprise programs,
- Build process standards that resemble six sigma or ISO programs,
- Establish upstream communication and fair redress,
- Conduct privacy impact assessments to objectively identify issues, problems and risks,
- Provide good reporting and disclosure tools to all stakeholders,
- Listen to customers about their privacy preferences, concerns and issues,
- Ensure both privacy goals and practical business objectives are met.

This practical understanding of what successful execution of privacy programs looked like from the inside would shape and focus my work over the next several years. As always, I seized opportunities to partner with thought leaders, captains of industry, privacy professionals, and government leaders. Through the 2000's, these opportunities were plentiful, and my office was busy!

---

[16] Information and Privacy Commissioner/Ontario and Deloitte &Touche, The Security-Privacy Paradox: Issues, Misconceptions and Strategies. (2004), `www.ipc.on.ca/images/Resources/sec-priv.pdf`

[17] Information and Privacy Commissioner/Ontario and Ponemon Institute, Cross-National Study of Canadian and U.S. Corporate Privacy Practices. `www.ipc.on.ca/images/Resources/cross.pdf`

We partnered, for example, with the Schulich School of Business at YorkUniversity on a paper entitled *Privacy and Boards of Directors: What You Don't Know Can Hurt You*,[18]which argued that privacy protection starts at the top and must have a C-suite level presence to provide real and effective organizational accountability. The paper outlined specific steps businesses should take, including conducting a self-assessment, educating staff about privacy, appointing a Chief Privacy Officer, making privacy an integral part of performance evaluations and compensation packages, executing regular privacy audits, and asking senior management the right questions about privacy.

We also contributed, in 2005, to an international business research syndicate, led by Don Tapscott, which was investigating the changing shape of businesses and ways in which to achieve new competitive advantages by adopting strategic information technologies into and business practices. The resulting paper[19] outlined the five major privacy challenges facing organizations for the next generation and offered solutions based on our *Privacy by Design* approach.

In 2009, we partnered with leading US firms to describe and illustrate how *Privacy by Design* could be applied to enhance organizational accountability to, and compliance with, international privacy laws and other requirements.[20] This publication was one of a series of joint papers applying *Privacy by Design* principles to illustrative concrete case studies in diverse areas such as remote health care[21], biometric systems[22], and the emerging "smart grid."[23]

Throughout the mid to late 2000's, there were countless speaking engagements, publications, partnerships, think pieces, web resources, and working relationships being built. During this period, my focus remained the same: to lay the foundation for protecting privacy well into the future by encouraging, entreating, and enabling organizations to implement *Privacy by Design*.

---

[18] Information and Privacy Commissioner/Ontario, Privacy and Boards of Directors: What You Don't Know Can Hurt You, http://www.ipc.on.ca/English/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=648

[19] Information and Privacy Commissioner/Ontario, Privacy and the Open-Networked Enterprise (2005), www.ipc.on.ca/images/Resources/priv-opennetw.pdf

[20] Ann Cavoukian, Ph.D., Marty Abrams, & Scott Taylor, Privacy by Design: Essential for Organizational Accountability and Strong Business Practices (2009), www.ipc.on.ca/images/Resources/pbd-accountability_HP_CIPL.pdf

[21] Information and Privacy Commissioner/Ontario & HP Canada, RFID and Privacy: Guidance for Health-Care Providers (2008), http://www.ipc.on.ca/images/Resources/up-1rfid_HealthCare.pdf

[22] Ann Cavoukian, Ph.D., and Alex Stoianov, Ph.D., Biometric Encryption: A Positive-Sum Technology that Achieves Strong Authentication, Security AND Privacy (2007), http://www.ipc.on.ca/images/Resources/bio-encryp.pdf

[23] Information and Privacy Commissioner/Ontario and The Future of Privacy Forum,Smart Privacy for the Smart Grid: Embedding Privacy into the Design of Electricity Conservation, http://www.ipc.on.ca/images/Resources/pbd-smartpriv-smartgrid.pdf; Information and Privacy Commissioner/Ontario, Hydro One, & Toronto Hydro, Privacy by Design: Achieving the Gold Standard in Data Protection for the Smart Grid, http://www.ipc.on.ca/images/Resources/achieve-goldstnd.pdf; and Information and Privacy Commissioner/Ontario, Hydro One, GE, IBM &TelventOperationalizing Privacy by Design: The Ontario Smart Grid Case Study,http://www.ipc.on.ca/images/Resources/pbd-ont-smartgrid-casestudy.pdf

2010: *Privacy by Design* Reaches a Tipping Point

By 2010, it was clear that the advocacy work of the past ten years was starting to pay off, as tremendous strides were made in evolving *PbD* from a conceptual framework into a practical one that was actually being applied by industry leaders. Organizations were no longer asking "why?" but "how?" The slide decks I had been using to present the business case for privacy went into retirement. They were replaced, instead, with slides about the growing momentum gathering behind *Privacy by Design*.

A high point for *PbD*, and for me personally, was the unanimous adoption of a landmark *Privacy by Design* resolution[24] by the full assembly of international Privacy Authorities and Regulators at the International Conference of Data Protection and Privacy Commissioners in Jerusalem.

The resolution recognizes *Privacy by Design* as an "essential component of fundamental privacy protection." It also:

- Encourages the adoption of the principles of *Privacy by Design* as part of an organization's default mode of operation; and
- Invites Data Protection and Privacy Commissioners to promote *Privacy by Design*, foster the incorporation of its Foundational Principles in privacy policy and legislation in their respective jurisdictions, and encourage research into *Privacy by Design*.

To help support that work, I released a *Privacy by Design* Curriculum.[25] The Curriculum provides resources that enable virtually anyone to understand and *teach others* about *Privacy by Design* and how its principles may be applied in particular settings.

Furthering the work of implementation, I involved my office in several groundbreaking projects in this field. One of them, a joint paper[26] with the Ontario Lottery and Gaming Corporation (OLG), focused on a very novel application of *PbD* in the field of Biometric Encryption.

2010 also saw significant privacy gains through the application of *PbD* in other arenas. Near the end of 2009, for example, my office worked closely with Google to develop a tip sheet on encrypting Gmail messages. Through that process, Google decided, in early 2010, to set the default so that it automatically encrypted all email messages sent by users of its Gmail service – a significant advancement!

We also did some work on implementing *PbD* in the hot-button area of targeted advertising. This kind of advertising brings with it a host of privacy issues, from those directly connected with the practice (e.g. the tracking of online behaviour, the use of location data as reported by mobile devices, etc.) to broader, Internet-wide topics (e.g. IP addresses as personal information, etc.).

---

[24] Information and Privacy Commissioner/Ontario, Landmark Resolution passed to preserve the Future of Privacy, `http://www.ipc.on.ca/images/Resources/2010-10-29-Resolution-e_1.pdf`

[25] All available at: `www.privacybydesign.ca`

[26] Information and Privacy Commissioner/Ontario, Privacy-Protective Facial Recognition: Biometric Encryption Proof of Concept, `http://www.ipc.on.ca/english/Resources/Discussion-Papers/Discussion-Papers-Summary/?id=1000`

In October 2010, we issued a joint paper[27] on one facet of the rapidly-evolving field of targeted advertising: precise IP geolocation, and the potential role of ISPs in the ad serving model. Our paper described the innovative technology developed by a highly innovative company, Bering Media, Inc., which allows ISPs to partner with an ad server to provide IP geolocation services without *any* disclosure of personally identifiable information about subscribers. Using their privacy architecture, the ISP can partner with an ad server without the server reading or modifying any packets travelling through the ISP's network.

While these and other *PbD* projects spanned widely disparate fields, all of them demonstrated the extent to which *Privacy by Design* fosters innovation by challenging system designers and engineers to think creatively. These projects have confirmed what I have long held to be true: rejecting the widespread but misguided view that privacy and other objectives are necessarily in conflict, opens up a *world* of possibilities.

The Long Road Ahead: Launching a Decade of *Privacy by Design*

At the start of 2011, the Future of Privacy Forum, a Washington-based think tank that promotes responsible data privacy practices, posted its First Annual List of Privacy Ins and Outs. I was delighted (and gratified) to see *PbD* make the list of what's "in."

2010 was an excellent year for *Privacy by Design*. We had clearly reached a tipping point. But this is not the time to rest on our laurels. There is much to look forward to, and there is yet much work to be done. Indeed, I am predicting that this year will launch the decade of *Privacy by Design*, and put in place a solid foundation for assuringthe future of privacy. Here are a few of the developments that I am hoping for and will be working towards in this decade:

1 PbD as a Fundmanetal Component of Privacy Frameworks

There is a growing momentum to enshrine the 7 Foundational Principles of *PbD* into privacy policies and regulatory frameworks. The U.S. Federal Trade Commission's noteworthy paper, *Protecting Consumer Privacy in an Era of Rapid Change: A Proposed Framework for Businesses and Policy Makers*,[28] named *PbD* as one its three recommendations.

Similarly, the European Commission's (EC) recent consultation paper[29] proposed *PbD* as a way to enhance the responsibilities of organizations. Peter Hustinx, European Data Protection Supervisor, has said "Privacy by Design needs to be explicitly

[27] Information and Privacy Commissioner/Ontario, Redesigning IP Geolocation: Privacy by Design and Online Targeted Advertising, `http://www.ipc.on.ca/images/Resources/pbd-ip-geo.pdf`

[28] Federal Trade Commission, Protecting Consumer Privacy in an Era of Rapid Change: A Proposed Framework for Business and Policymakers, `http://www.ftc.gov/opa/2010/12/privacyreport.shtm`

[29] Consultation on the Commission's comprehensive approach on personal data protection in the European Union, `http://ec.europa.eu/justice/news/consulting_public/news_consulting_0006_en.htm`

included as a general binding principle into the existing data protection legal framework. This would compel its implementation by data controllers and ICT designers and manufacturers while offering more legitimacy to enforcement authorities to require its effective application in practice...*Privacy by Design* should also be fully endorsed by the forthcoming European Digital Agenda and become a binding principle in future EU policies."[30]

In 2011, we are witnessing further movement toward embedding *PbD* into regulatory instruments, voluntary codes, and "best practices" all around the world. Among other things, this will significantly expand the understanding of how the principles of *PbD* may be interpreted in specific contexts, and applied to particular industries and technologies. And with the translation of the 7 Foundational Principles into 14 languages, *PbD* will truly become a global standard.[31]

2 PbD as a Fixture within Public and Private Sector Ecosystems

The signs are already beginning to appear: market leaders are starting to embrace *Privacy by Design*, and are, in turn, reaping the benefits. Recently, thought leaders Don Tapscott and Anthony D. Williams, authors of *Macrowikinomics: Rebooting Business and the World*, joined the ranks of strong voices in support of *PbD*, in an article[32] urging companies to adopt its principles. "Cavoukian's Privacy by Design playbook explains how to build privacy protections into everyday business practices. Every business needs to design privacy principles and practices into their operations."

Organizations that act proactively stand to gain a sustainable competitive advantage from their early adoption of responsible information practices, and enjoy savings of time and resources by building privacy in from the outset, rather than trying to retrofit an ill-fitting solution, after the fact.

3 A Generation of "Privacy Heroes"

Over the past few years, my office's annual *PbD* Challenge[33], our Developers Challenge (co-sponsored with Microsoft) and the *PbD* Ambassador program have begun to stimulate and recognize emerging leadership in the area of *Privacy by Design*. Armed with forward vision, technical expertise and respect for consumers and citizens, a committed pool of individuals and organizations, who we call "privacy heroes" – including researchers, academics, engineers, regulators, captains of industry, and privacy advocates – are emerging as forerunners in the implemention of *Privacy by Design*.

We look to these privacy heroes to expand the pool of *PbD* expertise, commitment, and innovation in 2011 and beyond, as the ranks of *PbD* supporters continue to swell.

---

[30] Privacy advisor calls for 'privacy by design' laws, `http://www.out-law.com/page-10851`

[31] The 7 Foundational Principles are being translated into a growing number of languages, including French, German, Italian, Spanish, Czech, Dutch, Estonian, Hebrew, Hindi, Chinese, Japanese, Arabic, Armenian, and Russian.

[32] Don Tapscottand Anthony D. Williams: Social media's unexpected threat, `www.ctv.ca/generic/generated/static/business/article1854656.html`

[33] Find information about the *PbD* Challenge at `http://www.privacybydesign.ca/events/upcoming-events/`

4 Innovative Applications of PbD

2010 saw *PbD* grow from a conceptual framework to a practical methodology that organizations are increasingly implementing. Significant projects in the areas of Smart Grid[34] and Privacy-Protective Biometric Facial Recognition[35], and mobile applications[36] marked the beginning of true innovation in applying the principles of *Privacy by Design*.

With market leaders like GE, HP, IBM, Microsoft, Oracle, Intel, Hydro One, the Ontario Lottery and Gaming Corporation, and new talent like Bering Media paving the way, 2011 promises to be a banner year for new and innovative applications of *PbD*.

5 Consistent Alignment between Business Practices and Consumer Expectations

Many organizations have lengthy, "legalistic" privacy policies that are difficult for consumers to read, let alone understand. Nonetheless, many consumers assume – incorrectly – the fact that a site posts a policy means that it will not share their personal information with unauthorized third parties. These expectations are certainly not well-founded, nor are they always consistent with current business practices.

Embedding privacy proactively will bring business practices into much better alignment with consumer expectations. While this process may take some time, I think we can look forward to seeing many positive steps in the coming year. And that will be good for everyone – consumers **and** businesses – because when consumers trust that their personal information is being protected, they will continue to support the growth of new forms of web-based commerce, without fearing for their information. Consumer confidence and business development – positive sum, win/win!

The road behind me may feel like a long one, but it has only just begun. Along the way, the cause of *Privacy by Design* has been greatly helped by our allies, partners, collaborators, and colleagues. Together we have fought to retain the ability to continue to enjoy privacy while enjoying the benefits of the modern age, win/win, not zero-sum.

The road ahead promises to be just as long. I invite all of you to join me in realizing the vision of a future world where privacy lives on by striving to implement innovative *Privacy by Design* solutions in your own organizations and lives. Our freedom and privacy may be at stake – what could be more important?

---

[34] Information and Privacy Commissioner/Ontario, Hydro One, & Toronto Hydro, Privacy by Design: Achieving the Gold Standard in Data Protection for the Smart Grid, http://www.ipc.on.ca/images/Resources/achieve-goldstnd.pdf and Information and Privacy Commissioner/Ontario, Hydro One, GE, IBM &TelventOperationalizing Privacy by Design: The Ontario Smart Grid Case Study, http://www.ipc.on.ca/images/Resources/pbd-ont-smartgrid-casestudy.pdf

[35] Information and Privacy Commissioner/Ontario and Ontario Lottery and Gaming Corporation, Privacy-Protective Facial Recognition: Biometric Encryption Proof of Concept, http://www.ipc.on.ca/images/Resources/pbd-olg-facial-recog.pdf

[36] Information and Privacy Commissioner/Ontario & Arizona State University's Privacy by Design Research Lab, The Roadmap for Privacy by Design in Mobile Communications: A Practical Tool for Developers, Service Providers, and Users. (2010), http://www.ipc.on.ca/images/Resources/pbd-asu-mobile.pdf

iSAM: An iPhone Stealth Airborne Malware

Dimitrios Damopoulos, Georgios Kambourakis, and Stefanos Gritzalis

Info-Sec-Lab Laboratory of Information and Communications Systems Security,
University of the Aegean, Samos, Greece
{ddamop,gkamb,sgritz}@aegean.gr
http://www.icsd.aegean.gr/info-sec-lab

Abstract. Modern and powerful mobile devices comprise an attractive target for any potential intruder or malicious code. The usual goal of an attack is to acquire users' sensitive data or compromise the device so as to use it as a stepping stone (or bot) to unleash a number of attacks to other targets. In this paper, we focus on the popular iPhone device. We create a new stealth and airborne malware namely iSAM able to wirelessly infect and self-propagate to iPhone devices. iSAM incorporates six different malware mechanisms, and is able to connect back to the iSAM bot master server to update its programming logic or to obey commands and unleash a synchronized attack. Our analysis unveils the internal mechanics of iSAM and discusses the way all iSAM components contribute towards achieving its goals. Although iSAM has been specifically designed for iPhone it can be easily modified to attack any iOS-based device.

Keywords: Malware, iPhone, iOS, Jailbreak, Stealth, Airborne, Rootkit.

1 Introduction

Mobile devices have evolved and experienced an immense popularity over the last few years. These devices have penetrated the market due to the variety of data services they offer, such as texting, emailing, browsing the Internet, documents editing, listening to music, watching videos and playing games in addition to the traditional voice services. As a result, analysts are expecting a mobile device population of 5 billion by 2015 [1]. Moreover, these devices are capable of performing sophisticated tasks and communicating through various wireless interfaces. As mobile devices hardware functionality and performance get improved, Operating Systems (OS) have similarly evolved. Modern mobile devices run sophisticated OS like Google Android, Apple iOS, Symbian, Palm OS, Blackberry RIM, Windows Mobile 7, that need to confront almost the same risks as desktop computers. It is thus apparent that this growth has exposed mobile devices to an increasing number of security threats. According to Chow and Jones [2], the only difference between desktop computers and mobile devices in terms of security risk is the challenge to understand the inner workings of the OS on different hardware processor architectures.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 17–28, 2011.

Very recently, Kaspersky Lab identified 39 new mobile malware families (SMS trojans, iPhone malware, Android spyware) with 143 variants [3] which try to compromise mobile device security. Also, according to a ScanSafe report, malware volume grew 300% in 2008, and it is noted that several of the legitimate web pages crawling on the Internet maybe infected by different kind of viruses [4]. In the same report it is stated that malicious image files comprised 10% of all Web malware encountered in 2009.

In this paper, we focus on iPhone device security. We create a smart malware namely iSAM to expose possible vulnerabilities of modern mobile devices and OS, and demonstrate that is relative easy to bypass any security control. Towards achieving its goals, iSAM employs a variety of programming techniques (pubic and private frameworks, override OS functions), backgrounding methods (daemons, dynamic libraries), as well as open source iPhone malware resources (e.g. Star exploit, iKee scanner logic). The aim of iSAM is to stealthily execute, six malware mechanisms, self-propagate wirelessly to other iPhone targets and finally connect back to the iSAM bot master server to update its programming logic or to obey commands and unleash a synchronized attack. Although specifically designed for iPhone 2G and iPhone 3G with the 3.1 and 4.0.1 iOS version respectively, iSAM can be easily adapted to attack other Apple iOS devices (iPhone 3GS/4 and all generations of iPod Touch). To the best of our knowledge this is the first rootkit-similar, airborne and stealth multifarious malware that is capable of infecting iPhone devices.

The rest of the paper is structured as follows. The next section presents previous work on the topic. Section 3 provides basic mobile malware design requirements and attributes. Section 4 describes the iSAM overall architecture and presents an analysis of the six proof-of-concept malicious iSAM's subroutines. The last session concludes the paper and gives pointers to future work.

2 Preliminaries and Related Work

Soon after the first iPhone was released, hardware and software modules were developed to bypass root privileges and overcome any restrictions. That is, only software signed by Apple's Certificate Authority is allowed to run on iPhone. This process is generally referred to as *"Jailbreak"*. Upon jailbreaking, the entire iPhone file system becomes open for use. Jailbreaking allows to create and execute third-party software without an official SDK from Apple. The first aim after jailbreaking was to bypass SIM-Lock. Specifically, every iPhone is locked to a particular network provider. Unlocking allows the user to place calls with any GSM/3G carrier by inserting a different SIM into the device.

The *installer* created by the development team RipDev, and *Cydia* created by J. Freeman were the first two package managers that allowed a user to browse and download third-party applications for jailbroken iPhones. The open-source Cydia became very popular after iPhone firmware version 2.0. Since then, every time a hacking team discovers a new iPhone exploit, they publish the corresponding software that jailbreaks the device. Also, the same software installs a version of Cydia, a SSH server, and enables the default root login password *"alpine"*.

In July 2007, T. Ormandy discovered *"libtiff"*, a buffer overflow method that has already been used to attack Sony's PSP device. Hackers inspected Apple's Mobile Safari web browser in order to test and take advantage of the same vulnerability that lay in the Tag Image File Format (TIFF) library, which is used for viewing TIFFs. Finally, they managed to successfully attack iPhone. Capitalizing on this vulnerability they created the web site jailbreakme.com. There, by selecting the "Slide-to-Unlock" button, a malicious TIFF file was simply opened from Mobile Safari leading to injection and execution of an arbitrary code and a straightforward Jailbreak. Once the iPhone has been jailbroken, the exploit patched the libtiff vulnerability in order to avoid future attacks. Apple patched this vulnerability with iOS 1.1.2 firmware. Vaibhav in his Project Report [6], discuses and analyzes the libtiff security breach in detail. Moreover, Chavez in [7] discusses how an intruder can successfully attack a network using a jailbroken iPhone. To perform the attack, she installs and uses a collection of powerful tools (e.g. Metasploit, Nmap, Whois, tcpdump, a terminal, WifiStumbler).

A year later, Apple introduce the new iPhone 3G that incorporates firmware version 2.0. Also, it offered a powerful Software Development Kit (SDK) that gave the opportunity to developers to create and deploy software under certain public frameworks so as to create the AppStore. In the end of July 2008, one of the iPhone third-party games namely *Aurora Feint* was removed from AppStore due to privacy concerns. Actually, the game was uploading to the developers server all contacts stored in the host iPhone. In 2009, serious privacy concerns appeared within the AppStore applications. MogoRoad and Storm8 are only two of the AppStore applications that have been removed after users' complaints about privacy concerns. In July 2009, users have raised serious concerns about their privacy in regard of the behavior of four tracking providers namely Pinch Media, Flurry, Medialets and Mobclix. J. Freeman tried to protect iPhone users by creating PrivaCy, an application for jailbroken iPhones, which blocks AppStore applications from tracking usage information.

The authors in [8] presented a vulnerability in SMS messages, which enables an attacker to inject fuzzed SMS messages into iPhones, Android and Windows Mobile devices. This vulnerability leads to a Denial-of-Service (DoS) attack remaining at the same time invisible to the service provider. This weakness was patched with the new 3.0.1 iOS firmware.

In 2009, researchers were trying to gain access to private information (i.e. contacts, photos, mails, SMS messages, passwords) stored in iPhone devices using various forensics methodologies. J. Zdziarski was the first one who using proper tools was able to retrieve unencrypted the full iPhone disk image. The same year he published a white paper with forensics techniques and tools that could be used to retrieve information from an iPhone device. During the same period, the first iPhone worm namely *Ikee* was released and a wave of worm attacks started. Ikee was simply changing the iPhone's wallpaper. Note that, Ikee was a self-propagating worm attacking only jailbroken iphones using the installed SSH server and the default root password. The same vulnerability was also used by *Dutch 5€ ransom*, a worm that locked the iPhone screen asking 5€ on a PayPal account in order to remove

the worm. *Privacy.A*, was another worm running in stealth mode and be able to steal personal data from the iPhone. In November 2009, a new highly disastrous version of Ikee, namely *iKee.B* appeared in several Europe countries. SRI International analysed iKee.B in [9] and provided technical details about the logic and the internal mechanics of the first iPhone Botnet. Although iKee.B acts similar to Ikee, it includes a Command & Control (C&C) logic to control all infected iPhones via a Lithuanian botnet server. Moreover, it is able to periodically update its malware behaviour. Finally, iKee.B, changes the default SSH password into "ohshit", and collects and sends all SMS messages stored in the device to the bot server. The iKee.B source code is published in [10].

Recently in [11] Seriot, presented some interesting attack scenarios on how a malicious application can use official and public frameworks, provided by Apple, to collect users private information (e.g., phone number, email account setting, keyboard cache entries, Mobile Safari searches and the most recent GPS location) programmatically. This happens without the user's knowledge and without being rejected by the AppStore review.

On July 2010, the United States government and the new Digital Millenium Copyright Act (DMCA) legislation announced that modifications of smartphones, like jailbreak or Unlock are legal as long as they obey the copyright law [12]. Based on the new law, in August 2010, Comex, an iPhone exploit developer, with the help of several other hackers introduced the exploit namely Star or JailbreakMe 2.0. This new exploit can jailbreak all Apple's products which incorporate iOS firmware versions from 3.1.2 to the current 4.0.1. Until then, all previous iOS firmwares have been jailbroken using offline exploits. Star, like JailbreakMe, is a remote browser-based jailbreak that uses two security flaws [13]. The first one, uses a corrupted font embedded in PDF files that crash the Compact Font Format (CFF) to allow arbitrary code execution, while the second one uses a vulnerability in kernel to escalate the code execution to unsandboxed root privileges. Any iOS mobile device that opens a jailbroken PDF file from a website, email, SMS, or Apple's iBook can be automatically jailbroken. A few days after Star was released, Comex published the source code [14].

3 Designing Principles and Requirements for iPhone

The primary aims of a smart malware is to infect the target, self-propagate to other targets and finally connect back to a bot master server. The latter action is highly desirable to update the malwares programming logic by improving already existed features and adding new ones, or to obey commands and unleash a synchronized attack. To achieve the affomentioned goals, the malware needs to fulfill some basic design requirements. First off, it needs to infect the device and gain root permissions. Also, it needs to run continuously in the background of the OS and has smart malware behaviour remaining stealthy to the legitimate user.

The only way to infect an iPhone and gain root permissions is by exploiting a vulnerability on an iOS jailbroken device. In case the target iPhone is already

jailbroken, the malware may attempt to use the SSH vulnerability[1] to wirelessly connect and infect the device. According to Cydia developer, J. Freeman, over 10% of the 50 million iPhones worldwide are jailbroken [15]. Although these devices constitute a large proportion for possible targets, it is necessary to find new ways to infect non-jailbroken iPhones.

To do so, we propose to create a malicious version of Star exploit [14] that is able to work wirelessly. As already mentioned, Star exploit consists of a PDF, which uses two security flaws allowing arbitrary code execution and gaining root privileges, and of a website *"JailbreakMe"* which stores the PDFs caring the exploits (one PDF for each iPhone version and one for each iOS version) [13]. Once the PDF is opened, a dynamic library (dylib) named "installui.dylib" provides graphic interface and downloads from the corresponding website a file named "wad.bin". After that it proceeds to jailbreak the iOS and install Cydia using a second dylib named "install.dylib". The file "wad.bin" is a binary file that contains any type of data; in this case it contains the "install.dylib" and the Cydia package. According to F-secure, any iOS mobile device that opens an exploited PDF file from a website, an email, an Apple's iBook application or accesses a website directly from an SMS message, can be jailbroken [16]. Note that iOS is capable of recognizing automatically hyperlinks sent via SMS.

Once a malicious Star PDF file is opened by an iPhone using our malicious Star version, it is being automatically jailbroaken and installed stealthily malicious software. Also, once an iPhone visits our website or opens the malicious PDF, the exploit procedure begins, stealthily, without providing any graphical interface or any information popups. Furthermore, we inject our malware into the "wad.bin". This means that once the jailbreaking procedure ends, Cydia and our malware will be both installed in the iPhone.

In order to create our malicious version of Star, it was necessary to modify the open source version of Star exploit [14]. Firstly, we decided to pack our malware as a Debian package. Once Cydia is installed in the iPhone, any file with the ".deb" extension stored in the folder "/var/root/Media/Cydia/AutoInstall", will be also automatically installed in the device. To inject our malicious package in the "wad.bin" file, it was necessary to modify the Star source class, named "install" and the python script "wad.py". Also, it was necessary to modify the source file "installui.m" which is used to build the dylib named "installui.dylib". In the source file "installui.m" we deactivated all displayed graphics interfaces making the exploit behave stealthily. Moreover, we edited the domain name from where our malicious "wad.bin" can be downloaded and we recalculated the size of our malicious "wad.bin" file editing the source where it was necessary. Last, after the installation of Cydia we shift our malware package into Cydia's auto-install directory. It is stressed that all these operations are possible because the Safari browsing process has acquired root access using the kernel bug.

The second requirement when designing our malware was the ability to run continuously in the background of the underlying OS. Until iOS version 4,

[1] The SSH vulnerability, allows intruders to remotely access a jailbroken device's file system using the SSH server and the default password "alpine".

multitasking was not officially supported. Unofficially, jailbroken iOS could support applications that run in the background as deamons or use Objective-C dylib. iOS being a Unix-based OS, can provide multitasking using *launchd*, a launch system that supports daemons and per-user agents as background-services. Once an iOS has been jailbroken, any installed application or shell script is able to behave as daemon by creating a launch plist and placing it into the "/Library/LaunchDaemons" iOS directory. Another way to support multitasking is with dylib. When an application is launched, the iOS kernel loads the application's code and data into the address space of a new process. At the same time, the kernel loads the dynamic loader i.e., "/System/MobileSubstrate/DynamicLibraries" into the process and passes control to it. In addition, it is possible to load a dylib at any time through Objective-C functions. Finally, from iOS version 4 and later, Apple provided seven APIs that allow applications to run in the background. Although these APIs are the native way for providing multitasking, it is not the best way to create and launch a malware. A program that uses the native way for backgrounding can be easily spoted by the user from the corresponding menu.

The last requirement is to design a smart malware that will remain stealthy and invisible to the user at all time. These smart malwares need to achieve their purpose stealthily by modifying OS code, functions and/or data. Officially, Apple does not provide any frameworks that override iOS functions. To fill the gap, J. Freeman has created and incorporated into Cydia MobileSubstrate extension, a framework that allows developers to deliver run-time patches to system functions using Objective-C dynamic libraries [17]. By creating a dylib, developers are able to build applications that run in the background and be able to replace internal system functions at the same time.

4 The iSAM Malware

Given the aforementioned requirements and possible solutions, we created iSAM. The iSAM malware has been implemented, using Objective-C source code compiled for iPhone ARM CPU. Also, iSAM was build using the unofficial ways (see Section 3) for backgrounding (daemons and dylibs), the public and private[2] frameworks and the MobileSubstrate framework with the *"substrate.h"* header that overrides iOS functions. This means that certain modules of iSAM can be classified as rookit.

iSAM consists of a main daemon written in Objective-C and combined with a proper launch plist (activated at device boot time) and six subroutines written as Objective-C functions, dylibs or shell scripts. The iSAM main daemon is responsible to manage all subroutines which are in charge of the propagation logic (iSAMScanner), the botnet control logic (iSAMUpdate) and the smart malware behaviour (iCollector, iSMSBomber, iDoSApp, iDosNet). iSAMScanner is activated during the device boot time and runs as a deamon in the background.

[2] Unsupported frameworks, which were retrieved directly from a jailbroken iPhone and have been dumped to get the headers.

iSAMUpdate is activated once per day and only if an Internet connection is available, while the rest four subroutines are activated once per week but at random times. Figure 1 depicts the overall iSAM architecture. Important pseudocode segments of all the iSAM subroutines discussed in this section can be found in [22].

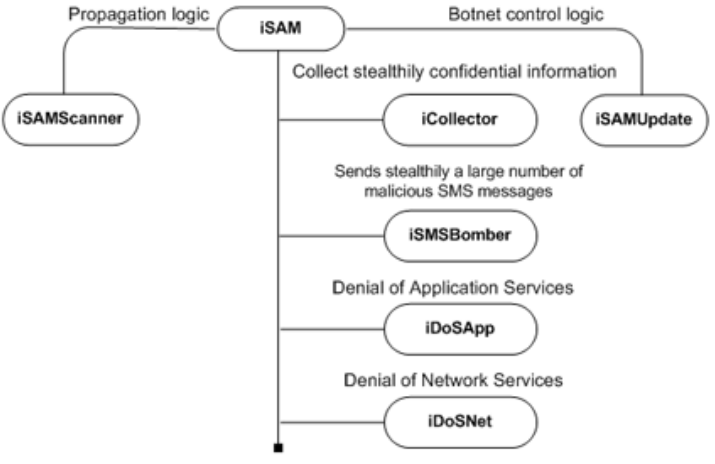


Fig. 1. iSAM architecture

In addition to iSAM, we setup a bot master server namely iSAM Server (iSAMS) having multiple functionality. iSAMS incorporates two basic modules: (a) a repository server where the newer or special customized version of iSAM is stored, and (b) a multithread socket server used to communicate with the infected devices to update iSAM program logic, to collect sensitive information and to control and execute commands directly on the iPhones. Also, iSAMS stores our malicious version of Star exploit namely *mStar*.

4.1 iSAM Infection Methods

As already mentioned iSAM uses two different methods to wirelessly attack and infect iPhone devices. The first method is by using iSAMScanner (see next section) which tries to detect jailbroken iPhones having the SSH vulnerability and infects them directly. Alternatively, we employ mStar, a modified version of the exploit Star, which is able to jailbreak the device and simultaneously infect it with iSAM. A recent report by F-Secure showed that nearly 79.8% of mobile phones infections were as a result of content downloaded from malicious websites or delivered by Bluetooth and SMS messages [18]. Capitalising on these results we use iSMSBomber (see section 4.5) as part of the second infection method to contaminate iPhone devices. iSMSBomber is able to read any telephone numbers stored in the device and send to them stealthly a SMS message with the domain of iSAMS. This is to trick the user into visiting iSAMS. In addition, mStar

can be delivered to an iPhone when visiting our iSAMS via a web link, email attachment or a legal popular AppStore application that uses a website link to redirect to iSAMS. Once 195.251.166.50 (iSAM.samos.icsd.gr) hyperlink is opened via a SMS message, mStar PDF is downloaded from iSAMS and loaded via Mobile Safari. After that, installui.dylib downloads wad.bin and install.dylib jailbreaks the iPhone and installs iSAM.

4.2 iSAMScanner: Scan, Connect, Infect

iSAMScanner is responsible for the propagation logic of iSAM. iSAMScanner driven by iSAM daemon, is activated at iPhone boot time. The iSAMScanner subroutine has three methods: *iScan, iConnect* and *Infector*. iScan is conducting three independent network scans just like iKee.B. Firstly, it scans iPhone's local WiFi network address space, then scans in a random way computer subnetworks on the Internet and finally scans a list of IP address range that belongs to a set of mobile phone companies in Greece (e.g. 195.167.65.0-195.167.65.255, GR, Cosmote) or in other European countries (e.g. 139.7.0.0-139.7.255.255, DE, Vodafone). When a vulnerable iPhone is detected, iConnect connects directly to the SSH Server using the default root password and by using Infector downloads the iSAM.deb package to the directory "/private/var/root/" of the target-device. Finally, Infector installs the package using the command dpkg -i –refuse-downgrade –skip-same-version iSAM.deb. From this step forward, the victim's device is under the control of iSAM.

4.3 iSAMUpdate: Update, Command, Control

iSAMUpdate, is responsible for the botnet control logic of iSAM. It is also used for connecting iSAM back to iSAMS to check whether a newer iSAM version is available. This allows iSAM to be updated e.g., with a new programming logic or follow commands directly from the server in order to unleash an attack. iSAMUpdate is connected back to iSAMS once every day as soon as an Internet connection is detected. Every time iSAMUpdate is activated, it retrieves some useful information from the device and sends them as a textmessage to iSAMS to be stored in the local database. The message is consisted of the iSAM version, the Unique Device Identifier (UDID), which is a unique serial number for each iPhone, the IP address from the e0 interface (WiFi connection on the iPhone) and the GPS coordinates, as long as a GPS is enabled. The following quintuplet gives an example of such a message {version016 ||3bdf7jc607h1j7te441sc02f5h5j6229db66hh63||62.217.70.167||26.700039||37.794186}. In case a newer iSAM version is detected, the server answers back with the name of this version, else it sends back a null message. It is not necessary for the server to respond with the latest version; instead it can answer with a customized response based on the UDID or the georgraphical coordinates if it wants to manipulate the phone in a special way or to attack devices selectively (e.g. attack all devices that roam to a certain area). Once the iSAM client receives the name of the version, it executes a Unix shell script named *"iUpdate.sh"* which is called

with the name of the version as a parameter. The shell script executes two script commands: the "curl -O iSam.samos.icsd.gr/debs/$1.deb", which downloads the newer iSam version directly from iSAMS and the "dpkg –i –refuse –downgrade –skip –same –version $1.deb", that uses the Debian package manager to install the new version. We should note that the name of the iSAM version, which the server has sent, is stored in the variable $1. It is also stressed that once the server has the client's IP address, is able to connect directly to the client's SSH service using the default root password.

An infected iPhone with iSAM is able to search for jailbroken iPhones into three different subnetworks (local subnet, random Internet subnet, mobile provider IP subnet) in order to infected them as well. Moreover, an infected iPhone can be updated or controlled by iSAMS. Lastly, if a non-jaibroken iPhone opens iSAM.samos.icsd.gr hyperlink through a SMS message, will get infected by mStar PDF.

4.4 iCollector: Gathers Private Information from the Device

The purpose of this attack module is to collect stealthily confidential information directly from the device. iPhone stores all user's data in SQLite databases and plist files without providing any encryption mechanism to secure their contents. Once an iPhone has been jailbroken, the iOS sandbox collapses and all databases and plists stored in the path "/var/mobile/Library/" are exposed to the attacker.

iCollector is an iSAM subroutine that collects stealthily sensitive information from iPhone's databases (call, sms, calendar, note) and from Safari's plist files (bookmarks and Web browsing history), storing them into a new database named iCollection.db. After the data collection takes place (see line #1-3 in [22]) and when an Internet connection is detected (line #3-6), iCollector is connected back to the iSAMS using the Client/Server model and TCP sockets in order to send the collected information (line #7-8). iCollector is a dylib written in Objective-C and uses an SQLite library to read, create and write to databases.

4.5 iSMSBomber: Sends Malicious SMS Messages in Stealth Mode

Like all GSM mobile devices, iPhone uses a set of commands, called AT (attention), to dial a number or exchange SMS messages. In addition to AT commands, iPhone employs a high level private framework, named "CoreTelephone" (incorporated to iOS), in order to communicate with the Baseband using Objective-C functions. However, this framework is neither available by the iOS SDK nor documented. The only way to overcome this issue is to retrieve the CoreTelephone framework directly from the files of a jailbroken iPhone and then use class-dump utility. Class-dump examines Objective-C runtime information stored in Mach-O files in order to generate the header files [19]. This procedure is necessary to execute every time a private framework is used. Once the CoreTelephone framework and the header files are available, a direct communication with the Baseband can be placed.

To take advantage of such a powerful framework, we create iSMSBomber. This is an iSAM dylib subroutine that sends silently say 1000 malicious SMS messages

using the private CoreTelephone framework and more specifically the CTMessageCenter header (line #1-3). Firstly, iSMSBomber makes an SQL query to the iPhone's address book database to retrieve telephone numbers from user contacts (line #4-5). In case no contact exists or the contacts are less than 1000, then random numbers are created to reach 1000. Every random number begins with the standard "003069" digit sequence, which represent a mobile phone number in Greece. Then iSMSBomber creates the following message: *"Hello, how are you? I have found an interesting website: 195.251.166.50 - Please send it to all'!'* and by using the sharedMessageCenter function, it sends the message to all the existing (plus random numbers if any) (line #6). Once an iPhone user receives this message and visits the website link, Mobile Safari web browser opens automatically and accesses the site. Recall that this domain is redirected to iSAMS that stores the mSTAR exploit, which in turn contains iSAM. Also note that this message is malicious only for iPhones iOS. Normally, once a SMS message is sent or received, automatically it is stored to the SMS database and a tone rings. iSMSBomber sends stealthily all the 1000 messages without storing them in the SMS database and without playing any tone. The only way to expose its presence is by the end of the month, when the mobile user receives his telephone bill assuming that the user does not usually send a high amount of messages.

4.6 iDoSApp: Denial of Application Services

Modern mobile devices are designed to increase the efficiency and the productivity of mobile users on the go. Therefore, by default, all mobile devices come bundled with some basic pre-installed applications or utilities. iPhone is offered with seventeen pre-installed applications. Additionally, AppStore contains more than $3 * 10^5$ iOS applications [20] offering the user the necessary on the go productivity. One of the main iOS applications is *SpringBoard* that manages the iOS home screen by displaying all icons of the available applications, starts the WindowServer and launches and bootstraps other applications [21]. For example, once a user touches the icon of an application, SpringBoard launches it. The goal of iDoSApp subroutine is to cause DoS in application launching by overriding some system functions required by SpringBoard.

In this context, iDoSApp is a dylib, which is activated at random time frames, is short-term (say 1-minute) and causes real DoS by non-loading an application. To achieve this, it is necessary to replace SpringBoard system functions, by class-dump SpringBoard in order to get the private headers and create a dylib. The headers used by iDoSApp are the *substrate.h* (used for overriding systems functions using the MobileSubstrate framework), the *SpringBoard.h* and the *SBApplicationIcon.h* headers (derived from the class-dump of SpringBoard (line #1-3)). SBApplicationIcon is a system function responsible for the behavior of all icons displayed by the SpringBoard. iDoSApp hooks, modifies and replaces SBApplicationIcon only when the selector is a launch message. A selector in Objective-C language is a message that can be sent to an object or class (4). Normally, every time the user touches on an application icon, a launch message is sent to SBApplicationIcon to load the application. In our case, once the iDoSApp

is activated, it blocks all launch messages that are sent to SBApplicationIcon causing DoS (line #5-7). iDoSApp will not compromise iSAM existence, as some applications can automatically close when an application is written for older or newer iOS versions or when they fail to manage the memory correctly.

4.7 iDoSNet: Denial of Network Services

The aim of iDoSNet subroutine is to cause DoS by deactivating for - say 30 seconds - all communication services (line #3-6). iSAM will activate iDoSNet at random times during a random day of the week. iDoSNet is using a private framework, namely Preferences.framework which can enable/disable the Airplane mode that controls 3G/GSM functions. Furthermore, iDoSNet uses the Apple80211. framework, a private framework that configures all 802.11 network interfaces, to cause DoS (line #1-2). We make the hypothesis that the duration of 30 seconds will not expose the existence of iSAM and the vast majority of users will suppose that it happend due to a temporary interruption to the wireless signal.

5 Conclusion

The evolution of malwares is a continuous race between intruders and defenders. Both use the same programming methods, tools and resources either to create a smart malware or to develop an intelligent malware detection mechanism. Overall, with the increasing risk of mobile malware, designing a highly secure mobile device is still a very challenging task. This paper concentrates on the very popular iPhone device. We design and implement iSAM a new multifunctional malware that is able to wirelessly infect and self-propagate to iPhone devices. iSAM is able to override OS functions and uses a variety of advanced programming methods (public and private frameworks), backgrounding methods (daemons, dynamic libraries), and open source iPhone malware resources (e.g. Star exploit, iKee scanner logic) towards achieving its goals. It is also able to hide its presence, and update its logic via the iSAM bot master server. iSAM incorporates six different malware mechanisms and utilises two different methods to wirelessly infect other devices. The purpose of our study is to highlight iOS weaknesses and offer in-depth information towards combating such threats.

Our future work will concentrate on obtaining detailed experimental results e.g. infection and untraceability rates, collector effectiveness etc as well as into modifying iSAM core so as to be able to automatically infect any iOS-based device.

References

1. Liu, L., Yan, G., Zhang, X., Chen, S.: VirusMeter: Preventing your cellphone from spies. In: Balzarotti, D. (ed.) RAID 2009. LNCS, vol. 5758, pp. 244–264. Springer, Heidelberg (2009)
2. Chow, G.W., Jones, A.: A framework for anomaly detection in OKL4-Linux based smartphones. In: Proceedings of the 6th Australian Information Security Management Conference (2008)

3. Kaspersky lab at mobile world congress 2009 in Barcelona, http://www.securelist.com/en/analysis/204792100/Kaspersky_Security_Bulletin_2009_Malware_Evolution_2009
4. Landesman, M.: The world's largest security analysis of real-world web traffic: annual global threat report, ScanSafe STAT, http://www.scansafe.com/downloads/gtr/2009_AGTR.pdf
5. Apple introduction to security overview, http://developer.apple.com/library/ios/‡documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html
6. Pandya, V.R.: iPhone security analysis. Project Report, Department of Computer Science, San Jose State University (2008)
7. Chavez, A.: A jailbroken iPhone can be a very powerfull weapon in the hands of an attacker. Project Report, Purdue University, Calumet's CIT Department (2008)
8. Miller, C., Mulliner, C.: Fuzzing the Phone in your Phone. In: BlackHat, USA (2009)
9. An analysis of the Ikee.B (Duh) iPhone botnet, http://mtc.sri.com/iPhone
10. iKee, http://vx.netlux.org/src_view.php?file=ikee.zip
11. Seriot, N.: iPhone Privacy. In: Black Hat, USA (2010)
12. Copyright, http://www.copyright.gov/1201
13. Technical analysis on iPhone jailbreaking, http://community.websense.com/blogs/securitylabs/archive/2010/08/06/technical-analysis-on-iphone-jailbreaking.aspx
14. Comex/Star, https://github.com/comex/star
15. The point of jailbreaking, http://www.saurik.com/id/12
16. How many ways can you remotely exploit an iPhone?, http://www.f-secure.com/weblog/archives/00002003.html
17. Mobilesubstrate, http://cydia.saurik.com/package/mobilesubstrate
18. The state of cell phone malware, http://www.usenix.org/events/sec07/tech/hypponen.pdf
19. Code the Code, http://www.codethecode.com/projects/class-dump
20. iTunes U downloads top 300 million, http://www.apple.com/pr/library/2010/08/24itunes.html
21. SpringBoard, http://www.iphonedevwiki.net/index.php/SpringBoard
22. iSAM: An iPhone Stealth Airborne Malware, Online Material, http://www.icsd.aegean.gr/postgraduates/ddamop/iSAM/iSAM.pdf

TCP Ack Storm DoS Attacks

Raz Abramov and Amir Herzberg

Bar Ilan University

Abstract. We present *Ack-storm DoS attacks*, a new family of DoS attacks exploiting a subtle design flaw in the core TCP specifications. The attacks can be launched by a very weak MitM attacker, which can only eavesdrop occasionally and spoof packets (a *Weakling in the Middle (WitM)*). The attacks can reach theoretically unlimited amplification; we measured amplification of over 400,000 against popular websites before aborting our trial attack.

Ack storm DoS attacks are practical. In fact, they are easy to deploy in large scale, especially considering the widespread availability of open wireless networks, allowing an attacker easy WitM abilities to thousands of connections. Storm attacks can be launched against the access network, e.g. blocking address to proxy web server, against web sites, or against the Internet backbone. Storm attacks work against TLS/SSL connections just as well as against unprotected TCP connections, but fails against IPsec or link-layer encrypted connections.

We show that Ack-storm DoS attacks can be easily prevented, by a simple fix to TCP, in either client or server, or using a packet-filtering firewall.

Keywords: Denial of service, TCP, secure network protocols.

1 Introduction

Most works in cryptography today adopt the all-powerful *Man In The Middle (MitM)* attacker model. The MitM attacker controls all of the traffic in the channels under him, with the ability to see, block and modify any package in the channel. In contrast, most works on Denial of Service (DoS) attacks, investigate the damage which much weaker attackers can cause, in order to focus on the most feasible and realistic attacks. Such weak attackers may only have the ability to send spoofed packets, or even weaker abilities - sending raw packets, sending only well-formed packets, or even merely issuing HTTP requests (e.g., puppets, see [5]).

In this work, we present and investigate the *Weakling In The Middle (WitM)* attacker model. The WitM attacker can eavesdrop on communication, but with significant limitations, mainly: eavesdropping only to one side of the connection, and receiving only a small percentage of the packets sent. These limitations are inspired by real-world wireless eavesdropping abilities, especially to open wireless networks; the 'one-sided' limitation is due to the fact that often the attacker is only able to eavesdrop to communication from the access point, and the low percentage is due to the weak reception by a remote eavesdropper. Open wireless

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 29–40, 2011.

networks are becoming more and more common, whether its in restaurants, malls or even as a city-wide infrastructure [9]. Attackers today can eavesdrop on public networks from a distance without the need for special equipment, and with poor reception quality (capturing low percentage of packets). It is widely known how to make a directional('Yagi') antenna, that can reach up to 12 miles of range and cost no more than a few dollars (see [2] or numerous web pages).

In addition to their limited eavesdropping capabilities, WitM attackers can also send spoofed packets to the network. This ability is very common, since many ISPs fail to properly deploy Ingress filtering [1]. However, we restrict the number of packets that the attacker can send into the network per attack; real attackers will try to restrict the number of packets they send, in order to stay hidden and avoid capture. Note also that sending few packets per attack increases the number of attacks that the attacker can perform simultaneously. These aspects are similar to the stealth attacker model of [3].

We present several *'Ack-Storm DoS Attacks'* that, by injecting (two or more) packets into an existing TCP connection, cause a long exchange of TCP packets between a client and a server, terminated only by connection reset or packet losses. This way enables a WitM attacker to disrupt services to local and regional junctions in the Internet infrastructure, as well as well as to individual web sites and services.

The Ack-storm behavior of TCP has been mentioned before in [4] and [10] as a side effect of TCP hijacking attacks, and thus as something to be minimized and prevented.

We present a typical scenario in Fig. 1, with a client Alice connected to an open wifi network AliceNet. Alice is connected to a remote web server Bob, over a standard TCP based connection, such as HTTP, SSL etc. The attacker

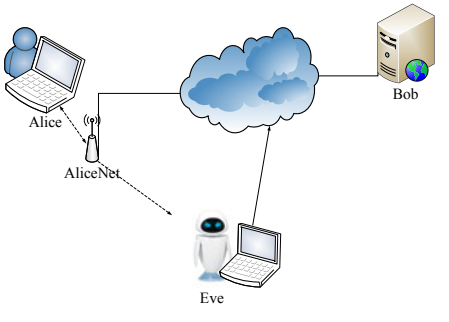


Fig. 1. Example attacker model - Alice is connected through the wireless access point AliceNet, to a remote web server Bob. Eve is able to receive occasional traffic from Alice's network. In addition, Eve's ISP does not filter traffic, so Eve is able to send spoofed packets to the Internet.

Attack	Highest Measured Ampl.	Ampl. Attacker
Two-Pkt Ack Storm	261,000	WitM
N Ack Storm	261,000	WitM
Everlasting Ack Storm	400,000	WitM
Opt Ack [7]	251.6	Client
Smurf [8]	≤ 1000	Spoofer
DNS Amp. [8]	73	Spoofer

Fig. 2. Comparing DoS Amplification Attacks

(Eve) has two abilities: eavesdropping and spoofing. Eve has a receiver antenna, with which she able to eavesdrop on (a small percentage of) packets sent by the access point over AliceNet. Eve in not able to inject packets into AliceNet, because of the long distance between them. Eve is also able to send raw packets into the Internet via its ISP. We assume that Eve's latency to both Alice and Bob is higher than the latency between them. Also, Eve cannot delay, drop, or otherwise affect any traffic sent in the network.

The Ack-storm attacks are based on the fact that, upon receiving a packet with the acknowledge number field (the receiver's sequence number) larger than the one sent by the receiving client, the client must, according to the TCP standard [6], resend the last sent acknowledgment packet to the other side, and discard the received packet. A design flaw in TCP causes the client and the server to be trapped in an infinite loop of sending and receiving empty acknowledgment packets.

The basic attack - Two Packets Ack Storm, as performed by the attacker, consists of three main stages:

1. Pick up (at least) one packet from a TCP connection between a client and a server.
2. Generate two packets, each addressed to one party and with sender address of the other party (i.e. spoofed). The packets must be inside the TCP windows of both sides. The packets should have content - at least one byte of data.
3. Send the packets to the client and the server at the same time. The connection will then enter an infinite loop of sending ack packets back and forth between both parties.

The *N-packets Ack Storm* attack and the *Everlasting Ack Storm* attack offer further amplification to the Two-packets Ack Storm attack, consuming more bandwidth and increasing the duration of the session. In our experiments, we measured an amplification factor of over 400,000, when performing the Everlasting Ack Storm attack - the highest amplification rate measured until today (see Fig. 2 for comparison with existing amplification attacks).

Contributions. This paper presents the following contributions:

1. We present the WitM attack model and demonstrate how a WitM attacker can preform DoS attacks.
2. We present the Ack-storm DoS attacks. These are powerful attacks, requiring low resources (low probability to intercept packets from the network, low bandwidth requirements) from the attacker and providing the highest amplification factor measured until today.
3. The Ack-storm DoS attacks demonstrate an advantage of the use of IPSec over the use of TLS/SSL. SSL connections are vulnerable to the Storm attacks, and even help the attacker target the servers and not the web proxy. IPSec, on the other hand, is immune to the attack, as it does not reveal TCP connection details to an eavesdropper attacker.

2 Two-Packets Ack-Storm Attack

In this section we present the flaw in the TCP standard that enables the Ack-storm DoS attacks, describe the Two-Packets Ack-Storm attack and explain the strengths and weaknesses of this attack.

2.1 The TCP RFC Flaw

The TCP RFC [6] defines all states and actions in a TCP connection. According to the RFC, the way to handle false data is, usually, to drop the packet. This behavior is recommended as it does not allow an attacker to trigger a response from either party by injecting false packets into the stream. However, there is one exception: When a TCP connection in ESTABLISHED state, and a packet is received with an ACK field that acknowledges data not yet sent, the client must act as follows (described in page 71 of the RFC):

1. Send an ACK (the last sent).
2. Stop processing ('drop') the segment. In particular, *ignore* the payload in the segment.

Sending an acknowledge packet in response to a malformed packet is not recommended - as such packet is clearly not a result of normal traffic. This behavior is what makes the Storm attacks possible.

2.2 Attack Description

To initiate the Two-Packets Ack-Storm the attacker sends two packets containing data: one to either side of a TCP connection. The attack uses the RFC flaw described above in order to cause the client and server to send false acknowledgment packets back and forth. No additional data could be sent once the attack takes place: every packet sent from now on will contain ack number higher then the one the receiving party has, and hence will only cause generation of another (malformed) ACK. If either side trys to send additional data over the channel (assuming the TCP send window is not full), the packets will increase the strength of the attack by creating additional 'sub sessions' of acknowledge packets (additional explanation can be found in the next section). Since according to the standard such packet must be dropped, and its data discarded, neither side can increase its sequence number, making it impossible for the sequence numbers to re-synchronize.

Figure 3 demonstrates the message passing between the client, server and attacker. By sending packets to both sides simultaneously, the attacker raises both the client and server's reserved ack numbers. This will make it impossible for them to overcome the false data sent, as all packets sent from that point will be considered false due to the ACK field being too high.

By sending a minimum of two packets, the Two-Packets Ack-Storm attack can cause hundreds of thousands of acknowledgment packets to be sent over a single TCP session. The figures, presented in the experiments section (Tab. 2)

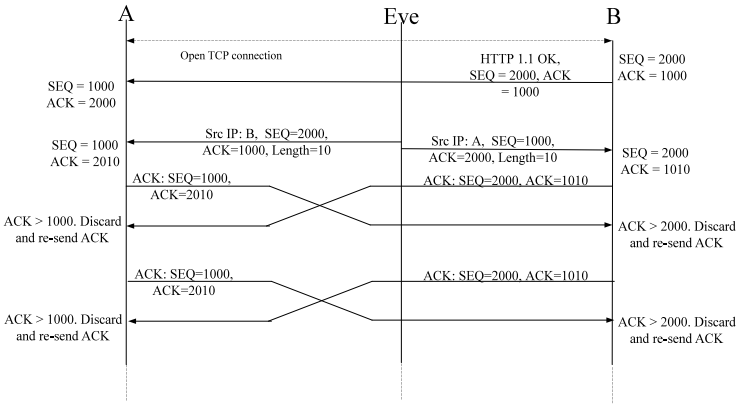


Fig. 3. The Two-Packets Ack-Storm attack. Numbers are for illustration only. The attacker sends both the client (impersonating as the server) and server (as the client) a message with length 10. Both sides send an ACK, while advancing their ACK number by 10. When the packets arrive at the other side, they contain an ACK field higher than the actual data sent. The client and server then send (according to the standard) the last ack sent by them, which triggers the loop all over again.

show that by sending only two packets, each with the minimal length of an Ethernet packet size (64 bytes), we can cause an amplification factor of over 261,000 times the original sending size.

The attack scenario is illustrated below (and in Fig. 3); to illustrate, we assume initial values of $A.SEQ = 1000(= B.ACK)$, and $B.SEQ = 2000(= A.ACK)$.

1. Eve sends A and B packets of length 10, each on behalf of the opposite side.
2. Upon receiving the packet, A advances A.ACK to be 2010, and sends an ack to B. B advances B.ACK to be 1010 and sends an ack to A.
3. When B receives a packet with $A.ACK = 2010$, when $B.SEQ = 2000$, he acts according to the standard: discards the packet and re-sends A the ack (in which $B.ACK = 1010 > A.SEQ$). A does the same, as it received a packet from B with $B.ACK = 1010$.
4. Both A and B receive packets with the ACK number bigger than their SEQ. The behavior in step 3 is performed again.
5. The loop continues when both parties keep receiving packets with an ACK larger than their sequence numbers, stopping only when both packets are dropped, or when one side reaches a timeout and ends the connection by RST.

2.3 Analysis

The attacker sends one packet to the client and one to the server, both with length of the minimum Ethernet packet size - 64 bytes. Each of the two packets sent by the attacker causes ACK packets (each of length 64 bytes) to be sent back and forth until the connection is terminated by the server, after $Time_R$ seconds. We are using the fact that the minimal length of an Ethernet packet is 64 bytes, while the size of

an empty TCP packet is only 60 bytes. This allows the attacker to send up to 4 bytes of data without adding to the length of the packet.

The two packets sent at the beginning of the attack are sent back and forth between the client and server, creating two 'sub sessions' of traveling packets. The attack continues until a RST packet terminates the attack after the maximal number R of retransmissions is sent; let $Time_r$ denote the time of the r^{th} retransmission (and $Time_R$ the time of the last retransmission before reset). During the total time of the attack, i.e., ($Time_R$), these two 'sub sessions' of packets would have caused a total of $128 \times \frac{R}{\rho}$ bytes sent, where ρ denotes the round trip time (RTT). Notice that the shorter ρ (the RTT between the client and server), the more effective the attack.

Since the attack interrupts an active session, altering the sequence numbers as it does so, acknowledgments of already sent packets are dropped. Therefore, the unacknowledged packets will be retransmitted by the sender. Since no retransmission will succeed, the sender will eventually give up and abort the connection.

Table 1 shows the retransmission scheme of Apache web server - the most common web server in the Internet today. Transmission times are all based on our experiments. From the table we can see that after $Time_R$ =225 seconds the server resets the connection. The server would have retransmitted ten times, in each the time waited between retransmissions is roughly doubled[1].

Table 1. Apache Server Retransmissions During the Attack

Retr. Attempt (r)	1	2	3	4	5	6	7	8	9	10	R=11
'sub sessions'	3	4	5	6	7	8	9	10	11	12	13
$Time_r(sec.)$	0	0.24	0.68	1.56	3.32	6.84	13.88	27.96	56.12	112.44	224.88
$Time_r - Time_{r-1}$ (sec)	0	0.24	0.44	0.88	1.76	3.52	7.04	14.08	28.16	56.32	112.44

Each retransmission packet that the server sends contains an acknowledgment number higher then the client's actual sequence number. Therefore, each retransmission attempt that the server sends starts a new 'sub session' of acknowledge packets sent between the client and server. Since the server makes 10 retransmission attempts, by the end of the last timeout there are 13 'sub sessions' of packets traveling back and forth. We mark $T_r = Time_r - Time_{r-1}$ as the current retransmission duration, ρ being the RTT, R the maximum retransmissions before connection abortion (for Apache R=11). The amplification factor that the attacker achieves, with the first retransmission sent roughly at the beginning of the attack, is therefore:

$$Amp_{Two}(R, T, \rho) = \frac{1}{2} \times \sum_{r=1}^{R} \left(\frac{T_r}{\rho} \times (r+2) \right) \tag{1}$$

[1] The retransmission policy of Microsoft IIS server is different: an IIS server will attempt a retransmission once every ten seconds, and initiate a connection abortion after sixteen unsuccessful retransmission attempts. In the analysis, we use the Apache retransmission properties, since it is more common, but the calculation could easily be modified to fit IIS (and other servers).

2.4 Experiments

In this section we present the results achieved both when we tested the Two-Packets Ack-Storm attack in the lab, and when we tested the attack on popular sites in the Internet. We tested the attacks on both HTTP and HTTPS sites, using both Apache and IIS web servers. The results from the experiments can be seen in Table 2. In the tests we measures the RTT to the site, the number of packets the attack generated and the time passed until the server reset the connection[2].

Table 2. Comparison of 2-Packet Storm Attacks on Different Sites

Site	Total Packets	Dur. (sec) ($Time_R$)	RTT (sec) (ρ)	Server Type	Total Bytes	Ampl.	Ampl. By Analysis
live (SSL)	13,000	229	0.270	IIS	832,000	7,500	5,002
oranim	20,000	120	0.180	IIS	1,280,000	10,000	16,000
yahoo	50,000	225	0.170	Ap.	3,200,000	25,000	25,415.5
facebook	60,000	225	0.160	Ap.	3,840,000	30,000	27,004
google	140,000	225	0.110	Ap.	8,960,000	70,000	39,299
bbc (uk)	190,000	225	0.060	Ap.	12,160,000	95,000	72,000
il.msn (p)	234,000	225	0.030	Ap.	14,976,000	117,000	144,098
bing (p)	320,000	225	0.018	Ap.	20,480,000	160,000	240,163.5
Lab	522,000	225	0.001	Ap.	33,408,000	261,000	4,322,944

In Tab. 2 we can see that while attacking sites running Apache, the time until termination of the attack remains constant - 225 seconds. This value is the connection abortion after maximum failed retransmissions. Once the attack takes place, no data is acknowledged in the session. That causes the server (usually the one sending the data over HTTP sessions) to retransmit the data 10 times, when each time the retransmission timeout doubles. Table 1 shows us the times of the retransmissions in relation to the beginning of the attack, and the termination of the connection occurred after the 10th timeout expired.

The Two Packet Storm attack presents a substantial amplification factor to the data sent by the attacker. The attack, however, has two main limitations:

1. The attack causes a constant number of 'sub sessions' , consuming a limited amount of network resources. If an attacker wishes to attack a high bandwidth target, he would have to use a large number of connections.
2. This attack is time limited, since a server will terminate the connection after reaching the maximum failed retransmissions attempts. After $Time_R$ seconds, the server will abort the connection, terminating the attack in the process. In order to attack a target for a time larger than $Time_R$, he would have to start new attacks to replace the old ones.

[2] We focused on Apache servers, because they are the most common. IIS and SSL(live.com) were tested for the attack but comparative research was not done on them.

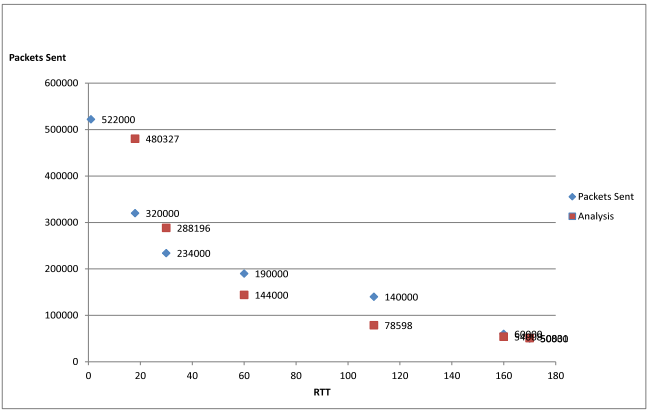


Fig. 4. Attack results when attacking various Apache-based sites ($Time_R = 225$), in comparison to analysis. We measured the number of packets sent and compared to the analysis. The graph shows the correlation between the analysis results, and the ones achieved in real attacks. Differences can occur due to packet loss/duplication, retransmission time deviations etc.

In the following sections we present two variations of the Two-packets Ack-storm DoS attack, which address the limitations described above.

3 The *N*-Packet Ack-Storm DoS Attack

The N-packet Ack-storm DoS attack enables the attacker to increase the number of packets sent over an attacked session. When using the N-packets Ack-storm DoS attack, the attacker can consume all of the bandwidth available for the session.

By Injecting additional ack packets to an attack already in progress, the attacker increases the bandwidth consumed by the attack. For every additional acknowledgment packet the attacker injects into the stream, another 'sub session' is created, and the client and server start passing the packet between them. This method bypasses the latency limitations of the client and server, since more 'sub sessions' simulate a shorter distance between the client and server: for the affective RTT to decrease by 0.5, the attacker doubles the number of 'sub sessions' in the connection.

3.1 Analysis

In order to maximize the effectiveness of the additional 'sub session' he creates, the attacker sends the additional Storm packets at the beginning of the attack. The number of 'sub sessions' at the end of the attack, for 2 packets sent in order to trigger the attack, N additional Storm packets and retransmission generated packets is $2 + N + R$. The amplification achieved would be, for R maximum retransmissions and 64 bytes acknowledgment packet length:

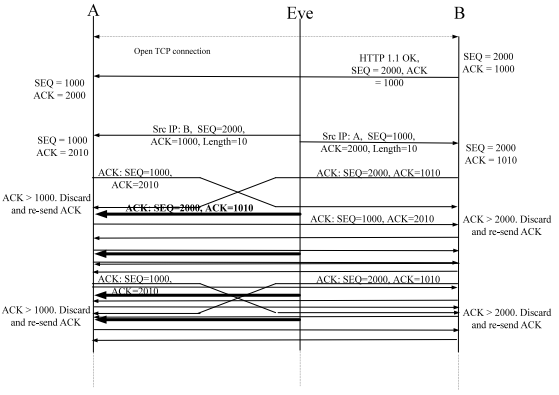


Fig. 5. The N-packets Ack-storm DoS attack. The storm packets sent by the attacker are in bold. Every new packet triggers another 'sub session' of acknowledge packets sent back and forth.

$$Amp_N\left(N,R,\rho\right)=\frac{1}{2+N}\times\sum_{r=1}^{R}\left(\frac{T_r}{\rho}\times\left(r+2+N\right)\right)$$

The attack minimizes the sessions needed to reach a high bandwidth consumption. Using storm packets, we were able to trigger *485,000* packets sent over a single session (with Yahoo.com), when the Two Packet Storm generated only 50,000.

When reaching the bandwidth limitations of a connection, packet losses will start to occur. Each packet loss will end the 'sub session' of that packet in the stream, while the rest of the 'sub sessions' will continue executing.

Table 3 presents the amplification results when attacking Yahoo.com. Number of packets increases as a function of packets sent, until reaches a point where packets losses start occurring (at about 42 Storm packets sent). Notice that packet losses do not stop the attack, but only limit the maximum bandwidth it can consume. We did not send over 42 packets in order to avoid causing real harm to users in the network, as would result from causing congestion on a live network).

Table 3. The N Packet Storm on Yahoo.com

Storm Packets Sent	Packets In Session	Attack Duration (T)	Bandwidth Consumption (Max.) $(\frac{Bytes}{Sec})$	Amplification Factor
2 (Two Packet Storm)	50,000	225	4,517	25,000
6	103,000	225	6,023	17,166
10	153,000	225	7,579	15,300
22	283,000	225	12,047	12,863
42	485,000	225	19,576	11,547

While bandwidth consumption increases as a function of the Storm packets sent, the amplification factor decreases. The attacker has to send an additional packet for every 'sub session' he wants to create, but the number of retransmissions (and the number of 'sub sessions' they create) remains the same. The disadvantage of lower amplification factor is balanced by the lack of need to manage multiple attacks in order to achieve the same amount of bandwidth usage.

4 Everlasting Ack Storm Attack

The main limitation of the two attacks presented so far is the the maximal connection duration $Time_R$. The attacker can avoid a connection abortion by artificially sending data over the channel, preventing the client or server from reaching a timeout. The Everlasting Ack Storm data packets should contain at least one byte of data, which will not add to their size because it is below the Ethernet packet size minimum. The packets should be sent at least every $Time_R$ seconds, in order to avoid the timeout. When data is being sent over the connection (however considered retransmission by both the client and server), the connection timeout is not triggered, allowing us to continue the attack until reset by application layer or any other network entity.

4.1 Analysis

The number of initial packets sent by the attacker is the same as the Two Ack Storm attack. The additional packets are sent every $Time_R$ seconds in order to maintain the connection. The amount of total data sent over the channel is composed of the initial $Time_R$ seconds, in which the amount of packets is identical to those of the Two Packet Storm attack, and the following minutes, in which the attack continues with the number of 'sub sessions' increases for every storm packet sent. The amplification achieved when sending Eve_R storm packets (for R maximum retransmissions of server):

$$Amp_{\infty}\left(R,T,\rho\right)=\frac{1}{Eve_R}\left(\times\sum_{r=1}^{R}\left(\frac{T_r}{\rho}\times r+2\right)+\sum_{r=R}^{Eve_R}\left(\frac{Time_R}{\rho}\times\left(r+2\right)\right)\right)$$

In Tab. 4 we show the different abilities the attacker can achieve by using the attacks described above. Notice that the amplification factor when sending storm packets without data decreases, but the total bandwidth increases as a function of the packets sent. Also notice that when sending storm packets with data, the attacker increases both the duration of the attack (by avoiding timeout), and the bandwidth consumed by it (as it opens another 'sub session')).

Using Storm packets with data, sent every 1 minute, we maintained an attack on Google.com for over 26 minutes, causing over 10,000,000 packets (over 640,000,000 bytes of data) to be sent before terminating the connection - an amplification factor of 400,000. We terminated the attack as part of our policy to avoid causing damage to the attacked sites.

Table 4. Amplification Types - Assuming No Bandwidth Limitation

Attack type	Data Sent	Attack Intervals	Attack Duration	Time Until Full Bandwidth Consumption	Ampl. Factor
Two	128 B	Once	$Time_R$	Never	Amp_{Two}
N	$128 \times Q$	Once	$Time_R$	Immediate	$Amp_N \leq Amp_{Two}$
Everl.	$128 \times L$	Every T_{rst}	$Time_R \times L$	$(Time_R) \times (\frac{BW}{64} - (2 + R))$	$Amp_\infty >> Amp_{Two}$

5 Preserving the Attack during Losses

When the attack reaches the bandwidth limitations of the channel, packet losses start to occur. For every dropped ack packet, one 'sub session' is stopped and the attacks consumed bandwidth drops by $\frac{64}{\rho}$Bps. In order to maintain the bandwidth of the attack, the attacker must generate an storm packet for every dropped ack packet.

When a TCP connection encounters losses, the TCP window size decreases rapidly. In the full version of the article, we present an experiment demonstrating the depredation in TCP bandwidth when performing an attack.

6 Conclusion and Future Work

In the article we presented the the WitM attacker model. We showed the Storm attacks, which can cause DoS to network infrastructure, as well as individual web sites and services. We showed both in analysis and in experiment results the high amplification factors the Storm attacks achieve.

The Storm attacks are just the beginning of a wide rage of attacks possible for the WitM attacker. Zombie-based DoS attacks can be accomplished by a WitM without the need to control the node itself, but only by using its connection details. Injection attacks are also possible to the WitM attacker, however the ability to inject the data in the correct timing requires further work, due to the high latency the attacker has. Finding additional DoS attacks and uses for a WitM attacker will add additional impact for the WitM attacker model.

Acknowledgments. Many thanks to Charlie Kaufman, Amit Klien and Ben Laurie for their important feedback and encouragement. Amit also introduced us to the earlier work discussing Ack storms (as an undesirable side-effect of TCP hijacking attacks), e.g. [4]

References

1. Borella, M., Grabelsky, D., Lo, J., Taniguchi, K.: Realm Specific IP: Protocol Specification. RFC 3103 (Experimental) (October 2001), http://www.ietf.org/rfc/rfc3103.txt

2. Chandra, P.: How To Make A WiFi Antenna Out of A Pringles Can. makeuseof.com (August 2009), http://www.makeuseof.com/tag/how-to-make-a-wifi-antenna-out-of-a-pringles-can-nb/
3. Herzberg, A., Shulman, H.: Stealth DoS attacks on secure channels. In: NDSS (March 2010)
4. Joncheray, L.: A simple active attack against TCP. In: Proceedings of the 5th Symposium on UNIX Security, pp. 7–20. USENIX Association, Berkeley (June 1995)
5. Lam, A., Akritidis, A.: Puppetnets: Misusing web browsers as a distributed attack infrastructure. In: SIGSAC: 13th ACM Conference on Computer and Communications Security. ACM SIGSAC (2006)
6. Postel, J.: Transmission Control Protocol. RFC 793 (Standard) (Sep 1981), http://www.ietf.org/rfc/rfc793.txt, updated by RFCs 1122, 3168
7. Sherwood, B.: Braud: Misbehaving TCP receivers can cause internet-wide congestion collapse. In: SIGSAC: 12th ACM Conference on Computer and Communications Security. ACM SIGSAC (2005)
8. Vaughn, R., Evron, G.: DNS amplification attacks. ISOTF (March 2006), http://www.isotf.org/news/DNS-Amplification-Attacks.pdf
9. Wong, M., Clement, A.: Sharing wireless internet in urban neighbourhoods. In: Steinfield, C., Pentland, B.T., Ackerman, M., Contractor, N. (eds.) Communities and Technologies 2007, pp. 275–294. Springer, London (2007), http://dx.doi.org/10.1007/978-1-84628-905-7_15, doi:10.1007/978-1-84628-905-7_15
10. Wu, B., Chen, J., Wu, J., Cardei, M.: A survey of attacks and countermeasures in mobile ad hoc networks. In: Xiao, Y., Shen, X.S., Du, D.Z. (eds.) Wireless Network Security. Signals and Communication Technology, pp. 103–135. Springer, US (2007), http://dx.doi.org/10.1007/978-0-387-33112-6_5, doi:10.1007/978-0-387-33112-6_5

Detecting Hidden Storage Side Channel Vulnerabilities in Networked Applications

Felix C. Freiling and Sebastian Schinzel*

University of Mannheim, Laboratory for Dependable Distributed Systems

Abstract. Side channels are communication channels that were not intended for communication and that accidentally leak information. A *storage* side channel leaks information through the *content* of the channel and not its *timing* behavior. Storage side channels are a large problem in networked applications since the output at the level of the protocol encoding (e.g., HTTP and HTML) often depends on data and control flow. We call such channels *hidden* because the output differences blend with the noise of the channel. Within a formal system model, we give a necessary and sufficient condition for such storage side channels to exist. Based on this condition, we develop a method to detect this kind of side channels. The method is based on systematic comparisons of network responses of web applications. We show that this method is useful in practice by exhibiting hidden storage side channels in three well-known web applications: Typo3, Postfix Admin, and Zenith Image Gallery.

1 Introduction

Covert Channels and Side Channels. A *covert* communication channel is a communication channel that was not intended to transfer information at all [15]. Today it is accepted that covert channels are impossible to avoid and hard to control. In practice, the sender P often does *not intend* to communicate with C but still leaks information. To distinguish this from the scenario of covert channels, the term *side channel* was used. Side channels abound and have been the focus of much research in areas like cryptographic hardware [11], API design [5] or non-electronic media [3].

In this paper we focus on side channels in applications running in the World Wide Web (web applications). Web applications are a prime communication mechanism today and side channels in web applications are relevant. Side channels are known as a special type of covert channel and covert channels are categorized into timing and storage channels. We thus adopt this discrimination for side channels and extend the general area of covert channels by distinguishing *storage side channels* from *timing side channels*. Whereas timing side channels are well researched in the World Wide Web, to our knowledge there is no well-founded research on the detection of storage side channels in web applications.

* Sebastian Schinzel was supported by Deutsche Forschungsgemeinschaft (DFG) as part of SPP 1496 "Reliably Secure Software Systems".

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 41–55, 2011.

To fill this gap, we give a general method by which storage side channels can be detected in web applications. As main example we show that in many existing web applications with user management it is possible to find out whether a certain high-privileged user account exists or not. This information is usually treated as confidential because knowledge of high-privileged user names eases password guessing and phishing attacks.

Side Channels in Web Applications: Related Work. Most related work has focused on constructing or detecting *timing* channels on the web (see for example Felten and Schneider [12], Bortz, Boneh and Nandy [7] or Nagami et al. [17]). A timing covert channel appears when one "process signals information to another by modulating its own use of system resources (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process" [18].

Only comparatively few authors have investigated possible *storage* channels on the web, i.e., channels that reside in the direct output of web applications. Bowyer [8] and Bauer [4] describe the possibility of hidden channels in HTTP. Bowyer suggests using superfluous URL parameters to form an upstream connection, and HTTP header fields or image steganography as downstream connection. Bauer chooses HTTP redirects, Cookies, referrer headers, HTML elements or active content to hide communication over the HTTP protocol. Kwecka [14] summarizes five different methods that were used to create covert channels within application layer headers: Reordering of headers, case modifications, use of optional fields and flags, adding a new field, and using linear white space characters.

We are aware of only two papers on the detection of hidden storage channels on the web. Chen et al. [9] detected a side channel in the packet size of encrypted traffic between browser and several web applications that process critical information such as healthcare information. Their target was to detect side channels that are visible for a man-in-the-middle who has access to the encrypted network traffic. Borders and Prakash [6] present a method to determine the maximum bandwidth of covert channels. They focus on covert channels where P is the web browser and C is the web server, which is the opposite direction of our approach. Their method neither detects, nor prevents covert channels, but focuses on determining the upper boundary of a covert channel's bandwidth. They analyze storage covert channels that enabled P to *purposefully* pass information to C. However, storage *side* channels appear if P *accidentally* and *unknowingly* passes information to C.

Detecting Storage Side Channels in Web Applications. In this paper, we exhibit a method for detecting storage side channels by comparing the differences in multiple responses. We call a storage side channel *hidden* if users will not detect it by observing only the visible elements of responses. In contrast to timing attacks, we measure differences in the *content* of the responses of web applications. The idea is to correlate the differences of web application responses with some secret information stored in the web server. Storage side channels are

noisy, i.e. if a user performs the same requests multiple times, many web applications will seemingly display the same content in the user's web browser. However, looking closely at the data of the web server's responses, one discovers that the responses differ slightly in many cases. We show that by a structured analysis it is possible to detect those differences that correlate with secret information, and are thus leaking information. Applying this method to three practical systems (Typo3, Postfix Admin, and Zenith Image Gallery), we were able to extract confidential information about existing user names and private images in public galleries.

Contributions. In this paper, we study deterministic hidden storage side channels in web applications and make the following three contributions:

- We formalize the context in which side channels in networked applications occur and identify necessary and sufficient conditions for hidden storage side channels to exist.
- We develop a general method to detect hidden storage side channels in networked applications and apply the method to web applications.
- We use our method to identify side channels in three common web applications: the popular content management system Typo3 [2], Postfix Admin [1], which is a web-based administration system for the popular email server Postfix, and Zenith Image Gallery [10], a web-based image gallery.

Outlook. The paper is structured as follows: In Sect. 2 we introduce a formal model of hidden storage side channels and give a necessary and sufficient condition for their existence. In Sect. 3 we derive a detection technique for such channels in networked applications and apply the method to three well-known web applications in Sect. 4. We conclude in Sect. 5.

2 Hidden Storage Side Channels in Networked Applications

We now abstract from concrete network protocols and investigate general networked applications and their susceptibility to side channels.

System Model. Our system model, depicted in Fig. 1, consists of an information producer P and an information consumer C. Both are modelled as *deterministic* state automata. Both P and C are connected through a shared resource R. In our case, R can be thought of as a network connection such as one using TCP over the Internet. Abstractly, R just allows to reliably transport a stream of bytes from P to C. The information that P sends to C can be any "high-level" digital object, e.g., a graph, a web page or a spreadsheet. P carefully constructed the high-level object in a way so that it only contains information that C is allowed to see. Producer P uses an encoding function φ that maps the high-level object into a bytestream suitable for transport over R. Similarly, C uses a decoding function

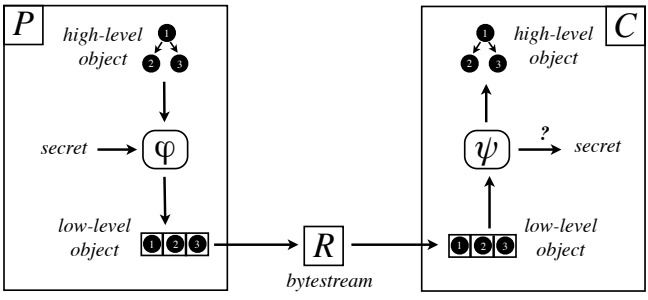


Fig. 1. System model

ψ that transforms the received bytestream into the same high-level object again. Note that P's high-level object is the same as C's high-level object.

Formally, let $\mathcal{H}$ denote the set of all abstract "high-level" objects and $\mathcal{L}$ denote the set of all allowed byte-sequences that can be transported over R. We assume that $\mathcal{L}$ is closed under subsequences, i.e., if $x \in \mathcal{L}$ and x' is a subsequence of x then $x' \in \mathcal{L}$ as well. We use $x \cdot y$ to denote the concatenation of two byte-sequences $x, y \in \mathcal{L}$.

Using this notation, φ is a function that maps elements from $\mathcal{H}$ to $\mathcal{L}$, and ψ is a function that maps elements from $\mathcal{L}$ to $\mathcal{H}$.

Definition of Hidden Storage Side Channels. A covert channel is a unidirectional communication channel that allows P to covertly communicate with C over R. P and C agreed on φ and ψ beforehand. The purpose of those channels is to hide the fact that P communicates with C over R.

A side channel is a special covert channel in which P *unintentionally* communicates with C over R. Side channels do not appear on purpose but appear accidentally during system construction. C's challenge is to discover and decode the side channel in order to get the information from P.

Intuitively, a storage side channel allows C to infer a secret value from the content that P sent. A hidden storage side channel is a storage side channel that can only be detected in low-level values from $\mathcal{L}$, as the high-level values are identical so that an observer of $\mathcal{H}$ will not detect the differences.

In general, we assume that the decoding function ψ is the inverse of the encoding function φ, i.e., $\forall h \in \mathcal{H} : \psi(\varphi(h)) = h$. However, many encoding formats in practice contain redundancy in the sense that the same high level objects can be encoded differently by φ. This can cause hidden storage side-channels, as we now explain.

Abstractly, a storage side channel regarding some secret information exists if P encodes some object $h \in \mathcal{H}$ dependent on some secret (see Fig. 1). For example, if the secret consists of just one bit, P could encode h into l_0 if the bit is 0, and into $l_1 \neq l_0$ if the bit is 1. If both l_0 and l_1 will be decoded into the same high-level object h again, hidden storage side channels appear. Investigating the encoding of l_0 and l_1 reveals the secret.

A Necessary and Sufficient Requirement. Note that our system model subsumes the scenario of Kemmerer [13] since P and C are connected through a direct communication link. Therefore, it is clear that side channels may exist. However, we refine Kemmerer's conditions to our case and derive a sufficient requirement for hidden storage side channels to exist in our system model.

It is sufficient for storage side channels to exist in our system model if there exist ψ-synonyms in $\mathcal{L}$, i.e. there exist distinct values l_1 and l_2 in $\mathcal{L}$ that ψ maps into the same value in $\mathcal{H}$, formally:

$$\exists l_1, l_2 \in \mathcal{L} : l_1 \neq l_2 \wedge \psi(l_1) = \psi(l_2)$$

To prove that this is a sufficient condition we have to show how a hidden storage side channel can be constructed. This is rather straightforward, as explained above: Depending on the secret s, the producer (possibly unintentionally) selects either l_1 (for $s = 0$) or l_2 (for $s = 1$). By investigating the low-level encoding of the message, i.e., before applying ψ, the consumer can learn the value of s.

We now argue that the condition of ψ-synonyms is also a necessary condition. For this we have to prove that the existence of hidden storage side channels implies that ψ-synonyms hold on the encoding.

So assume that we have a system in which hidden storage side channels exist. From the definition of *storage* side channel and in contrast to timing side channel, we know that information is transported using the *content* of messages exchanged over R. Since the channel is hidden, the channel is not visible on the level of $\mathcal{H}$.

There are two cases to consider:

1. The content directly reflects secret information, i.e., a value l_1 implies $s = 0$ and l_2 implies $s = 1$. In this case, we directly have the condition of ψ-synonyms.
2. The content reflects secret information via the order of byte sequences exchanged over R. In the simplest case, this means that there are two byte sequences l and l' such that $l \cdot l'$ encodes $s = 0$ and $l' \cdot l$ encodes $s = 1$. Note that $\psi(l \cdot l')$ must be equal to $\psi(l' \cdot l)$. In this case, choose $l_1 = l \cdot l'$ and $l_2 = l' \cdot l$ and the condition of ψ-synonyms holds again.

So overall, if hidden storage side channels exist, then the requirement of ψ-synonyms must hold. Therefore, the requirement is not only sufficient but also necessary.

Attacker Model. We assume that the attacker is in full control over the the information consumer. The goal of the attacker is to deduce the secret that resides in the information producer P. The allowed interactions between the attacker and P must conform to the protocol used on the shared resource (e.g., HTTP), i.e., we do not consider protocol-specific attacks.

3 Detecting Storage Side Channels in Networked Applications

We now study the question, how to detect the existence of storage side channels for a given networked application. This is a non-trivial task since the dependency

of the low-level encoding on the secret may not be known or may be time or context dependent.

Secret-dependent and Secret-independent Differences. Consider the case where P leaks secret information over a storage side channel to C and this secret consists of one bit s. Now assume two possible worlds a and b that are identical in all aspects except the value of the secret bit s: In world a we have $s = 0$ and in world b we have $s = 1$. P now sends a message via R to C. Let A denote the message sent in world a, and let B denote the message sent in world b. If a and b are identical except for the values of s, then any difference between A and B is a direct influence of the difference of the secret value. (Recall that both P and C are deterministic automata.) Hence, measuring the differences between A and B yields *secret-dependent* differences in messages.

Ideally, we want to identify secret-dependent differences since they can be used to infer the value of s. However, in practice, it is difficult to construct two identical worlds a and b in which P has absolutely the same state (even using virtualization). To make our method more robust and practical, we allow observing messages at different points in time as shown in Fig. 2. If we send two successive messages A_0 and A_1 in world a, for example, then we can measure the differences between these two messages. Since $s = 0$ for both messages, any differences between A_0 and A_1 must be *secret-independent*. Secret-independent differences can result from different times or relative orders of the sending of A_0 and A_1. Now assume that a message in world a is sent at time t_0 and another message in world b is sent at time t_1. Let A_0 denote the former and B_1 denote the latter message. If we compare A_0 and B_1, we find a composition of secret-dependent and secret-independent differences (see Fig. 2). The challenge is to identify the secret-dependent differences and associate them with particular

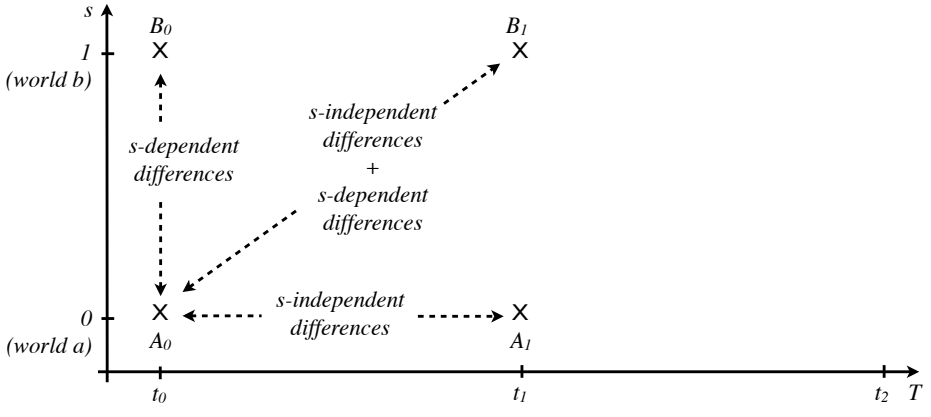


Fig. 2. Realistic measurements yield a mix of secret-dependent and secret-independent differences. The challenge is to reliably extract secret-dependent differences from the mix.

values of s. In the following section, we describe our method that we used to successfully uncover storage side channels in web applications.

Definitions and Notation. Let A be a byte sequence $A = a_0, a_1, a_2, \ldots$. We define an *edit* e *of* A to be a tuple (p, q) such that

1. $p \in N : 0 \leq p < |A|$ and
2. $q \in N : 0 \leq q \leq |A| - p$.

Intuitively, an edit describes a "change" of the byte sequence, more specifically a removal of q bytes starting at position p. We formalize this by defining what it means to apply an edit to a byte sequence.

Let $e = (p, q)$ be an edit of A. The *application of* e *to* A, denoted $A|_e$ is the byte sequence resulting from removing q bytes starting at index p in A, formally:

$$A|_e = a_0, a_1, \ldots, a_{p-1}, a_{p+q}, a_{p+q+1}, \ldots$$

Here are some examples: Assume A is the byte sequence "01234". Then $A|_{0,2} = 234$, $A|_{1,3} = 04$, $A|_{4,1} = 0123$, $A|_{2,1} = 0134$ and $A|_{1,0} = 01234$. In contrast, both $A|_{2,4}$ and $A|_{5,0}$ are undefined because both $(2, 4)$ and $(5, 0)$ are not edits of A.

We now describe what it means to apply multiple edits to A. We say that an *edit* (p, q) *is compatible with edit* (p', q') iff (if and only if) $p + q < p'$. Intuitively, two edits are compatible if they affect different parts of A. They are not compatible (incompatible) if they overlap. For example, for $A = 01234$, edit $(1, 3)$ and $(4, 1)$ are compatible whereas $(1, 4)$ and $(4, 1)$ are incompatible.

Let $\mathcal{E}$ be the set of all possible edits of A and $E = e_1, e_2, e_3, \ldots, e_n$ be a sequence of n edits of A such that for all i, $0 < i < n$, e_i is compatible with e_{i+1}. The *application of* E *to* A, denoted $A|_E$, is defined as:

$$(\ldots((A|_{e_n})|_{e_{n-1}})|_{e_{n-2}}\ldots)|_{e_1}$$

Note that the definition applies edits in reverse order, i.e., starting with e_n. This is not a necessary requirement as edits could be applied in any order. However, it simplifies the definition since index numbers for subsequent edits $e_{n-1}, e_{n-2}, \ldots$ remain the same.

In the next step, we use the notion of edits to define the longest common subsequence and a special difference operator Δ. Let A and B be two byte sequences. The byte sequence x is a *common subsequence (CS) of* A *and* B iff there exists two sequences of compatible edits E_A and E_B such that $A|_{E_A} = x$ and $B|_{E_B} = x$. In other words, x can be constructed from A by applying E_A to A and x can be constructed from B by applying E_B to B.

The byte sequence x is a *longest common subsequence (LCS) of* A *and* B if x is a common subsequence of A and B and $|x|$ is maximal. We denote this by writing $x = LCS(A, B)$.

Note that there can be multiple LCSs for pairs of byte sequences. For example, for $A = 212$ and $B = 121$ there are two longest common subsequences, namely 21 and 12. In our implementation, we chose to return the "leftmost" LCS of the left parameter, i.e. $LCS(212, 121) = 21$ and $LCS(121, 212) = 12$.

We are now ready to define the "difference" operator Δ that will be important in our method to detect storage side channels. Let A and B be two byte sequences. The operation $A \Delta B$ results in a compatible sequence of edits E of A such that $A|_E = LCS(A, B)$. In other words, E is the sequence of edits needed to produce out of A a longest common subsequence of A and B. For example, let $A = 212$ and $B = 121$, then $A \Delta B = (2, 1)$ and $A|_{(2,1)} = 21$.

The Difference Algorithm. In practice, secret-independent differences often result from dynamic values like session identifiers or time stamps. Note that for time dependent values, the time period during which the measurements were taken is important. For example, if the day field of a time stamp changes only after 24 hours (once every day), we will not be able to observe this dependency if all measurements are taken within the same day. For two such byte strings A_1 and A_2, we must expect $|LCS(A_1, A_2)| > 0$, i.e., even two randomly generated byte strings are likely to have an LCS with non-zero length. So the idea is to incrementally compute the LCS of a *sequence* of byte strings $A_1, A_2, A_3, \ldots$, i.e., first $LCS(A_1, A_2)$, then $LCS(LCS(A_1, A_2), A_3)$, then $LCS(LCS(LCS(A_1, A_2), A_3), A_4)$, etc. At some point, say while going from A_i to A_{i+1}, the length of the resulting byte sequence will not change any more. This means that all random elements have been eliminated from all A_i, i.e., we have computed the "static core" of the response sequence A_i.

In the following, we will now generalize our observations into an algorithm to identify secret-dependent differences. Fig. 3 shows a graphical representation of the steps.

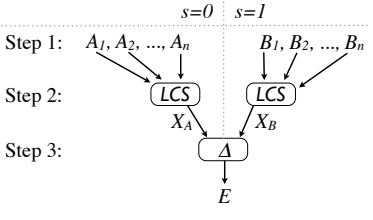


Fig. 3. The difference algorithm

1. Record n responses with $s = 0$ (denoted $A_1, A_2, \ldots, A_n$) and n responses with $s = 1$ (denoted $B_1, B_2, \ldots, B_n$). The value of n depends on how many responses are required to detect all dynamic parts in the response's content. Section 4 introduces empirical values of n for different applications.
2. For all $1 \leq i \leq n$ recursively calculate

$$X_A = \begin{cases} A_1 & \text{for } i = 1 \\ A_i|_{A_i \Delta X_{i-1}} & \text{for } 1 < i \leq n \end{cases}$$

Compute X_B similarly using responses B_i. Intuitively, the byte sequences X_A and X_B correspond to the "static" parts of A_i and B_i, respectively, in which any secret-independent differences are removed. Finding the secret-independent differences E_A of a response A_1 is straight forward: $E_A = X_A \Delta A_1$.

3. Now compute $E = X_A \ \Delta \ X_B$. The set E contains an approximation of all secret-dependent differences. If $E \neq \emptyset$ the probability of a storage side channel is high.

Looking closely at the edits in E it is often possible to decode the storage side channel, as we show in the following sections where we apply this method to real-world web applications.

4 Application of Method to HTTP/HTML

We now apply our method to web applications in which we wish to detect storage side channels. Such side channels can be, for example, the information whether a certain user account exists in a web application. Another example is the fact whether certain data items (such as images) are stored in a database. We start by verifying whether the condition of ψ-synonyms is satisfied in HTTP/HTML. Then, the attacker has to perform two steps. First, the attacker needs to find out whether the targeted application leaks the required information over a storage channel. Second, given that the targeted application is vulnerable to storage side channels, the attacker creates a search pattern. If the search pattern is applied to a single response, the attacker can find out the secret.

Storage Side Channels in HTTP and HTML. We now apply our system model to HTTP and HTML, and define that P is the web server, C is the web client, and R is the TCP channel between C and P. The set of high-level objects $\mathcal{H}$ consists of all web pages in the form of bitmaps displayed in C's web browser. The set of low-level objects $\mathcal{L}$ is the set of all ASCII sequences sent over R. Therefore, φ is the function used by P that "turns" a web page's intended bitmap into a valid HTML document and attaches it to a HTTP response before it is sent via R to C. The function ψ is the parser in C's web browser that reads the content of the HTTP response, creates the page's bitmap, and displays it on the screen of C.

HTTP responses contain a set of common headers, one of which is the *Server* header. The value of this data field is usually set to the particular type and version of the web server. However, this value is almost never used by the client and can therefore be set arbitrarily without affecting the web page displayed in C's web browser. Other examples for synonyms come from the fact that HTTP header fields are not case sensitive, i.e. "date" and "dATe" denote the same header field name. Moreover, HTML allows including meta information (e.g., author) in the HTML header which can be set to arbitrary values without affecting the content displayed in the browser.

To summarize, HTML as well as HTTP allow synonymous values and are therefore potentially vulnerable to storage side channels.

Detecting Storage Side Channels in Web Applications. As a precondition for our first attack scenario, the attacker needs to know at least two existing user names and two non-existent user names in one instance of the targeted software.

The attacker performs n login requests with known to be valid user names and records the responses A_i. All login requests in this scenario will fail because the attacker does not know any valid password in the target system. Thus, the system will always return an error message. He then calculates X_A from all responses and observes the decreasing length of the static part. In parallel, the attacker performs n login requests with *invalid* user names and records the responses (B_i). Again, these login attempts will fail because the user name does not exist. Then, he calculates X_B. n was large enough if the LCS of the responses stays constant for larger n.

The attacker then calculates the set of edits (E) representing secret-dependent differences. If $E \neq \emptyset$, the application leaks information about whether a user name exists through a storage side channel.

For our second attack scenario, the attacker needs to have access to at least one gallery containing no private pictures (A) and another gallery with at least one private picture (B). The attacker records n responses of A and n responses of B and calculates X_A, X_B, and E. If $E \neq \emptyset$, the application leaks information about whether private images exist in a gallery through a storage side channel.

For the implementation of our algorithm, we chose Myers' LCS algorithm [16]. It has near linear time and space requirements with regard to the size of the two responses that need be compared. A Java implementation of the algorithm performed around 70 LCS operations per second on responses with 12 Kilobytes.

Exploiting Storage Side Channels in Web Applications. The attacker now creates a search pattern from the secret-dependent edits in E. The application of this pattern to the response of a login request to any instance of the vulnerable web application determines whether the login request was performed using a valid or invalid user name.

The calculation of E was a one time effort. From now on, the attacker can learn about the existence of a chosen user name with a single request and on any system on the web which runs the vulnerable software. The quality of the results is independent of the network conditions and of the amount of changed bytes in the secret-dependent difference.

In the following, we apply our method to three real-world examples.

Example 1: Storage Side Channels in Typo3. In our first example, we analyze whether the user authentication logic of the backend administration interface of Typo3 [2] version 3.6.0rc2-0.0.1, an open source and well-known content management system (CMS), is vulnerable of storage side channels. For this, we installed a copy of Typo3 in our lab. By default, Typo3 already creates one administrative user account (*admin*) during installation. It is possible to delete or rename this user name after installation, but we can assume that not all web administrators do this. Although we performed the measurements in our lab, this measurement could have been performed at any Typo3 instance on the web.

Let s be the boolean value denoting whether a user name exists. We now record a set of responses from login attempts with valid user names and another

set with invalid user names. We then generate the "static core" of both sets, namely X_A and X_B. The set of edits $E = X_A \Delta X_B$ yields the set of secret-dependent differences.

Interestingly, the different sizes of X_A and X_B correlates with s and therefore suggests that the analyzed functionality is vulnerable of storage side channels. Unfortunately, the size of the response will most certainly differ among several installations because of site-specific customizations of the web page and the web server type and configuration. An attacker would therefore need to analyze the response size for each target before performing the actual attack, which makes the attack easier to detect. Furthermore, if the web page contains secret-independent dynamic data with differing size, e.g. from a news ticker or changing ads, the response size is dynamic as well, which would require additional probabilistic analysis and even more measurements. Our method is immune to these disturbances as it produces the exact positions of secret-dependent differences, which allows the creation of search patterns for this particular change.

Fig. 4 shows the positions of secret-independent differences in the byte stream of X_A. It is apparent that some values in the header, presumably time stamps, and one position in the HTML body change independently of s. Fig. 5 shows secret-dependent differences which form a storage side channel. This storage side channel leaks the existence of a given administrative user name. Note, that our method distinguishes secret-independent differences and secret-dependent differences even if they overlap, as it is the case here. It seems that the differences are located in the HTTP headers, because the positions of the secret-dependent differences are far to the left.

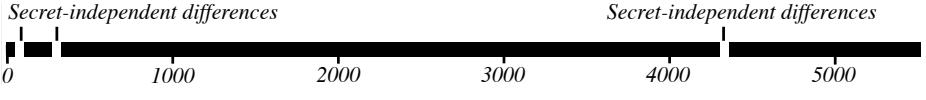


Fig. 4. The positions of secret-independent differences in a response of Typo3

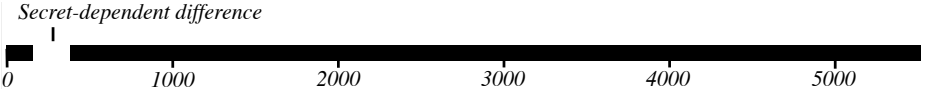


Fig. 5. The positions of secret-dependent differences in a response of Typo3

Fig. 6 shows a direct comparison of the HTTP headers generated with $s = 0$ and $s = 1$, respectively. The secret-dependent differences are highlighted in both responses. A manual analysis of these exposed secret-dependent differences yields that not only the values of the HTTP headers in Typo3 change depending on s but also the order in which the headers are declared. Note that the content sizes of both responses are equal, i.e. the change of response size we observed earlier in this section does not affect the HTML content but solely happens in the HTTP header.

Non-existent user name (s=0)	*Existing user name (s=1)*
HTTP/1.1 200 OK *Date: Mon, 25 Jan 2010 11:47:55 GMT* *Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny4 with Suhosin-Patch* *X-Powered-By: PHP/5.2.6-1+lenny4* *Expires: **Thu, 19 Nov 1981 08:52:00 GMT*** ***Last-Modified: Mon, 25 Jan 2010 11:47:55 GMT*** ***Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0*** ***Pragma: no-cache*** *Vary: Accept-Encoding* *Content-Type: text/html;charset=iso-8859-1* *Content-Length: 5472*	*HTTP/1.1 200 OK* *Date: Mon, 25 Jan 2010 11:47:45 GMT* *Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny4 with Suhosin-Patch* *X-Powered-By: PHP/5.2.6-1+lenny4* *Expires: **0*** ***Cache-Control: no-cache, must-revalidate*** ***Pragma: no-cache*** ***Last-Modified: Mon, 25 Jan 2010 11:47:45 GMT*** *Vary: Accept-Encoding* *Content-Type: text/html;charset=iso-8859-1* *Content-Length: 5472*

Fig. 6. Secret-dependent changes in Typo3's HTTP header that depend on whether a user name exists

Example 2: Storage Side Channels in Postfix Admin. In our second example, we analyze whether the user authentication logic of Postfix Admin [1] version 2.3 rc7, an open-source Email administration frontend [1], is vulnerable of storage side channels. We calculated X_A, X_B, and E as described previously. After roughly $n = 15$ comparisons, the sizes of X_A and X_B do not change any more.

Fig. 7 shows the positions of the secret-dependent and the secret-independent differences in the responses. Fig. 8 shows a direct comparison of the secret-dependent differences in the HTTP headers and in the HTML content generated with $s = 0$ and $s = 1$, respectively. The secret-dependent differences are highlighted in both responses. A manual analysis of the exposed secret-dependent differences yields that Postfix Admin automatically fills a user name in the form field of the response if and only if the user name exists in the database and is an administrative user. Strictly speaking, by the definition given in this paper, this vulnerability does not qualify as a *hidden* storage side channel, as it *can* be observed in the browser. In practice, however, it is unlikely that this storage side channel is detected in a manual security test, because of the subtle nature of the difference. Actually, common web browsers may remember user names that a user filled in the same login form earlier, in order to automatically fill in the same user name for subsequent requests. We thus want to highlight, that our method will also detect "visible" storage side channels that would probably slip through most manual security analysis.

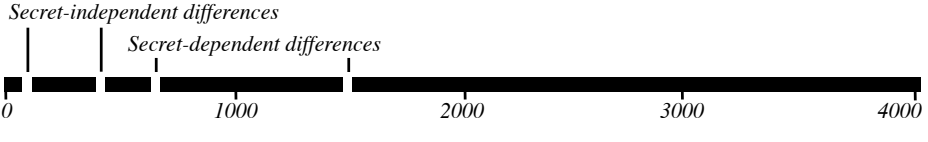


Fig. 7. The positions of secret-dependent and secret-independent differences in a response of Postfix Admin

Non-existent user name (s=0)

```
...
Content-Length: 3612
...
  <tr>
    <td>Login (email):</td>
    <td><input class="flat" type="text" name="fUsername"
                value="" /></td>
  </tr>
```

Existing user name (s=1)

```
...
Content-Length: 3626
...
  <tr>
    <td>Login (email):</td>
    <td><input class="flat" type="text" name="fUsername"
                value="admin@admin.de" /></td>
  </tr>
```

Fig. 8. Secret-dependent changes in Postfix Admin's HTTP header and HTML content that depend on whether a user name exists

Example 3: Storage Side Channels in Zenith Image Gallery. In our last example, we analyze whether the Zenith Image Gallery [10] version v0.9.4 DEV, a well-known open source picture gallery, is vulnerable of storage side channels. Zenith allows administrators to upload pictures, which anonymous users can view. Administrators can also delete pictures from the gallery or mark them as private pictures. Private pictures are still shown to administrators but are hidden for anonymous users. Here, we analyze if an attacker can find out the existence of private pictures in the public gallery.

Let s be the boolean value denoting whether private pictures exist in a chosen gallery with seven public images and one additional private picture. We then record a set of responses B from this gallery with the additional private picture ($s = 1$) and another set A from the same gallery in which the private picture is deleted ($s = 0$). The measurements are performed as anonymous users and only 7 pictures are visible during both measurements. We then calculate X_A, X_B, and E. After roughly $n = 11$ comparisons, the sizes of X_A and X_B do not change any more.

Fig. 9 shows a comparison of the HTML content of a gallery web page. Interestingly, only a single number depends on s. A-responses (no private images) have a constant size of 12460 bytes. B-responses (1 private image) have the same size with 12460 bytes. A-responses as well as B-responses have 35

7 public images, 0 private image (s=0)

```
<div style='float:left'>Pictures -
  <a href='display.php?t=bycat&amp;q=4&amp;nr=7&amp;st=0&amp;upto=12&amp;p=1'>
    <span style='color:#fff'>Other</span>
  </a>
</div>
```

7 public images, 1 private image (s=1)

```
<div style='float:left'>Pictures -
  <a href='display.php?t=bycat&amp;q=4&amp;nr=8&amp;st=0&amp;upto=12&amp;p=1'>
    <span style='color:#fff'>Other</span>
  </a>
</div>
```

Fig. 9. The secret-dependent difference in Zenith's HTML content of a gallery that depends on the existence of private pictures in a gallery

secret-independent bytes. Only a single byte forms the hidden storage side channel. A manual analysis yields that the secret-dependent number represents the sum of public image and private images. In case of $s = 1$, the number is 8 as there are 7 public pictures plus 1 private picture in the gallery. Thus, the hidden storage side channel not only leaks the fact that there are private pictures in a gallery, but also the amount of private pictures in the gallery.

5 Conclusions

The success of our detection method critically depends on the secret that is to be observed and therefore it is hard to automate. This makes it extremely difficult to quantitatively assess the significance of side channel vulnerabilities in web applications in general. For this paper, we analyzed 15 applications for *hidden* storage side channels and quickly identified the three examples documented here.

In principle, it is rather straightforward to avoid storage side channels by avoiding the condition of ψ-synonyms that we identified in this paper. This can be done, for example, by "fixing" the encoding function φ so that there are no synonymous values in $\mathcal{L}$. In reality, however, the encoding scheme is often given and, for the case of HTTP and HTML, it is not feasible to remove the general existence of synonymous values.

So in practice, more pragmatic approaches are necessary, such as trying to remove the relation between s and the choice of the synonymous values, to make the inference from $\mathcal{L}$ objects to the value of s (and vice versa) impossible. This approach, however, largely depends on the vulnerable application and on how this application chooses synonymous values depending on s. In practice, therefore, side channels in web applications will have to be treated in a similar way as covert channels in other systems: Educate developers and give them effective techniques for detecting such channels in their systems. Furthermore, future protocol designers should carefully avoid conditions in which synonymous values are acceptable.

Acknowledgments. We thank Martin Johns for the helpful comments on a previous version of this paper.

References

1. Admin, P.: Web based administration interface (2010), http://postfixadmin.sourceforge.net/
2. The TYPO3 Association: Typo3 content management system (2010), http://www.typo3.org/
3. Backes, M., Dürmuth, M., Unruh, D.: Compromising reflections-or-how to read LCD monitors around the corner. In: IEEE Symposium on Security and Privacy, pp. 158–169. IEEE Computer Society, Los Alamitos (2008)
4. Bauer, M.: New covert channels in HTTP. CoRR, cs.CR/0404054 (2004)
5. Bond, M., Anderson, R.: API-level attacks on embedded systems. Computer 34(10), 67–75 (2001)

6. Borders, K., Prakash, A.: Quantifying information leaks in outbound web traffic. In: Proceedings of the IEEE Symposium on Security and Privacy (May 2009)
7. Bortz, A., Boneh, D.: Exposing private information by timing web applications. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) WWW, pp. 621–628. ACM, New York (2007)
8. Bowyer, L.: Firewall bypass via protocol stenography (2002), http://web.archive.org/web/20021207163949/, http://networkpenetration.com/protocol_steg.html
9. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-channel leaks in web applications: a reality today, a challenge tomorrow, Oakland, CA. IEEE, Los Alamitos (May 2010)
10. CyberiaPC.com. Zenith picture gallery (2007), http://zenithpg.sourceforge.net/
11. European Network of Excellence (ECRYPT). The Side Channel Cryptanalysis Lounge. Internet (April 2010), http://www.crypto.rub.de/en_sclounge.html
12. Felten, E.W., Schneider, M.A.: Timing attacks on web privacy. In: SIGSAC: 7th ACM Conference on Computer and Communications Security. ACM SIGSAC (2000)
13. Kemmerer, R.A.: Shared resource matrix methodology: An approach to identifying storage and timing channels. ACM Transactions on Computer Systems 1(3), 256–277 (1983)
14. Kwecka, Z.: Application layer covert channel - analysis and detection (2006), http://www.buchananweb.co.uk/zk.pdf
15. Lampson, B.W.: A note on the confinement problem. ACM 16(10), 613–615 (1973)
16. Myers, E.W.: An $O(ND)$ difference algorithm and its variations. Algorithmica 1(2), 251–266 (1986)
17. Nagami, Y., Miyamoto, D., Hazeyama, H., Kadobayashi, Y.: An independent evaluation of web timing attack and its countermeasure. In: Third International Conference an Availability, Reliability and Security (ARES), pp. 1319–1324. IEEE Computer Society, Los Alamitos (2008)
18. Department of Defense Standard: Department of Defense Trusted Computer System Evaluation Criteria (December 1985)

Breaking reCAPTCHA: A Holistic Approach via Shape Recognition

Paul Baecher, Niklas Büscher, Marc Fischlin, and Benjamin Milde

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. CAPTCHAs are small puzzles which should be easily solvable by human beings but hard to solve for computers. They build a security cornerstone of the modern Internet service landscape, deployed in essentially any kind of login service, allowing to distinguish authorized human beings from automated attacks. One of the most popular and successful systems today is reCAPTCHA. As many other systems, reCAPTCHA is based on distorted images of words, where the distortion system evolves over time and determines different generations of the system. In this work, we analyze three recent generations of reCAPTCHA and present an algorithm that is capable of solving at least 5% of the challenges generated by these versions. We achieve this by applying a specialized variant of shape contexts proposed by Belongie et al. to match entire words at once. In order to handle the ellipse shaped distortions employed in one of the generations, we propose a machine learning algorithm that virtually eliminates the distortion. Finally, an improved shape matching strategy allows us to use word dictionaries of a reasonable size (with approximately 20,000 entries).

1 Introduction

CAPTCHAs—Completely Automated Public Turing tests to tell Computers and Humans Apart—are used to prevent automated use of online services. Typically, this is a challenge/response protocol where the user is, for example, required to read and submit a heavily distorted image of a word. While there exists an unmanageable number of such systems, upon closer examination most of them turn out to be insecure against automated solvers. This usually happens as soon as the system is used on a popular website and exposed to many users. Since CAPTCHAs are omnipresent and constitute an integral security mechanism in today's Internet services, this situation is highly unsatisfying. It is thus even more important to investigate promising systems and determine the level of security they provide.

The reCAPTCHA System. In contrast to the vast number of broken schemes, one particular implementation, reCAPTCHA [1], has been successfully in use for several years now. Two distinct key features are seemingly responsible for this comparatively long lifespan. First, the generation algorithm of reCAPTCHA challenges is proprietary and not public, meaning that challenges are provided

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 56–67, 2011.

via a centralized infrastructure. This allows reCAPTCHA to adjust their system at any given time, for example in the event of a successful attack on the system. Since such adjustments immediately affect all users of reCAPTCHA, no legacy installations exist that could still be prone to the attack. Moreover, since the algorithm is kept secret, it is tedious to analyze the variance of the challenges. Second, every challenge is guaranteed to have a minimum level of resistance against common OCR techniques. This is due to the way challenges are generated: instead of artificially rendered and distorted characters, reCAPTCHA uses words on which two OCR systems failed; a by-product of digitizing huge amounts of text.

The answer to such challenges is thus inherently unknown to the system. In order to verify the user's response, reCAPTCHA follows a statistic approach and presents two words in each challenge. One word is the unknown *scan word*, the other word is a known *verification word*. As long as the user provides the correct answer for the verification word, the response is considered correct and the given solution to the scan word is recorded. It is important to observe that the solution to the scan word, when viewed separately, is not relevant to pass the test. This is a critical detail when it comes to estimating success rates. Ideally, both classes of words should be indistinguishable, but this is not the case. For instance, it is entirely possible to have an algorithm that reliably recognizes scan words but performs poorly on verification words. Clearly, such an algorithm will not be suitable to break the system.

The centralized system makes it also hard to analyze the security of reCAPTCHA, since there are no public distinct and explicit versions of the generation algorithm. Hence, subtle modifications and revisions of the algorithm are not necessarily visible to the user. It is, however, possible to identify a set of major generations as shown in Figure 1. In the first generation, for example, words are struck through with a horizontal line; the third generation adds inverted ellipse-shaped blobs. Although the challenges of the second and fourth generation look similar, the distortion of the latter is more regular and exhibits a compact mathematical description. Furthermore, it seems that the fourth generation also uses less common words which tend to be excluded from dictionaries.

In this work, we focus on the security of the third and fourth[1] generations of reCAPTCHA.

Our Contributions. We present an implementation to break the latest generation of reCAPTCHA using shape contexts [2]. As opposed to previous work in this area, our algorithm is quite efficient with reasonably sized dictionaries of 20,000 words (i.e., shapes). To our best knowledge, this is also the first attempt to break reCAPTCHA using shape contexts and, in particular, to do this in a holistic fashion where entire words are matched atomically at once. Since reCAPTCHA is based upon the hardness of character recognition our results may therefore also stimulate new approaches for OCR.

In order to attack the third generation of reCAPTCHA, which includes an ellipse-shaped distortion object, we propose a machine learning framework that

[1] This is the latest version of reCAPTCHA as of November 2010.

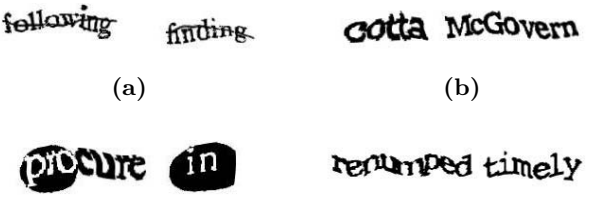
is able to detect and remove this distortion almost entirely. This allows us to treat challenges from the second to the fourth generation uniformly with one algorithm since the challenges of these generations are then sufficiently similar. Finally, we employ a novel method to quickly match a given query shape against a large list of dictionary words. This is done by taking the first and last character of the challenge word into account (which are considerably easier to segment) and then subsequently reducing the search space logarithmically.

2 Related Work

Attacks on reCAPTCHA. As mentioned before, reCAPTCHA has been immune against major attacks for a long time. Wilkins [16] announced in 2009 to have broken the first generation in Figure 1 of the reCAPTCHA system with a success rate of 5% (conservative estimate) to 17.5% (optimistic estimate) in early 2008. The two types of estimates stem from the fact that reCAPTCHA offers two classes of words for which it only knows the solution to the verification words. In 5% of the cases Wilkins got both words on his 200 test data right, in another 25% he got only one word correctly. Making the optimistic assumption that in half of these cases this is indeed the verification word yields the bound of 17.5%.

Wilkins essentially uses three techniques for his attack: the distortion line is removed by applying some combination of erode/dilate matrices, then he runs an OCR program, and finally he uses a dictionary to make a guess for the word. This is iterated for several matrices and the most likely answer about all these runs is output. Wilkins also ran tests again the second generation of reCAPTCHA (without the distortion line). Here, he was able to solve in a data set of 40 about 5% of the puzzles, simply running an OCR program. He concluded that the new system should be even weaker than the previous generation but this claim seems to be hard to formally back up by the restricted experiments.

At DefCon 2010, and independently of our work, Houck [7] announced successful attacks on the third generation reCAPTCHA (with the ellipse). He estimates to achieve a success rate of about 10%, based on experiments on about 5,000 CAPTCHAs. However, this optimistic estimate is again based on the assumption that, in about 75% of the cases, the solution for one word only is indeed for the verification word. Houck's approach is based on removing the ellipse,



(a) (b) (c) (d)

Fig. 1. Major generations of reCAPTCHA, in chronological order

segmenting the word into characters via "dips" in the upper margin—making this attack fundamentally different from our holistic approach—and running an OCR program.

Soon after these attacks became public, reCAPTCHA changed to the fourth generation. Houck also briefly discussed extensions of his attack to this generation, yielding an estimated success rate of 30%. In this sense, reCATPCHA became actually weaker.

Segmentation vs. Recognition. Text-based CAPTCHAs—the overwhelming majority of today's systems—present strings of letters and digits, possibly forming words of a natural language. These strings are rendered onto a rastered image and presented to the user. The exact process of how strings are rendered is crucial to the security of the system. Specifically, it is vital that subsequent characters overlap not only by their bounding box, but also touch each other such that the string forms one large connected component. This is absolutely necessary for security because, as Chellapilla et al. discuss in [5], *recognizing* single characters is a solved problem where computers even outperform humans. The task of *segmentation* on the other hand, where one is interested in partitioning a string in terms of a connected component into its individual characters, is still considered fairly resistant against automated attacks [4].

Shape Contexts. Recognition and comparison of shapes is a recurring problem in computer vision. Typically, shapes are transformed to a compact feature vector or descriptor which allows for fast comparison under slight variations of shapes. Belongie et al. propose a descriptor called *shape contexts* in [2] and efficient retrieval methods in [12]. A shape is described by a set of histograms, where each histogram corresponds to the distribution of vectors from one contour point to all other contour points. Shape matching against the database is then done by matching the sets of histograms of two shapes, where histograms are compared with a χ^2-metric. Clearly, this technique can be used to match characters and digits in a CAPTCHA. In [14] Mori and Malik successfully attack the EZ-Gimpy CAPTCHA using shape contexts; their approach is able to match the correct word in 93% of the time. Lladós et al. investigate this technique to spot words in historical documents from a predefined set of keywords in [9]. Although this can be viewed as a direct application to the OCR task, it is not designed to digitize entire documents. Rather, they are interested in metadata extraction by looking for specific words. Unfortunately, they do not mention the size or order of magnitude of their reference dictionary.

3 Our Techniques

In this section, we present our framework to break the recent reCAPTCHA generations 2–4. Our system can be divided roughly into two phases. In an offline "learning" phase, we create synthetic challenges based on a dictionary of English words. Each challenge is transformed to a descriptor that consists of a set of shape context histograms. We then create a database that contains all histograms for all words in the dictionary. A given (real) challenge in the online

phase is transformed exactly the same way and the resulting histograms are matched against the database; the closest match is the output of our algorithm.

Note that this basic version of our attack operates on entire words only, thus circumventing the task of segmentation. This technique is commonly known as holistic word recognition [6,10,11,8]. One can interpret this as a recognition task on a large alphabet, i.e., where entire words are the letters.

Figure 2 gives an overview of the transformation process from challenge images to descriptors.

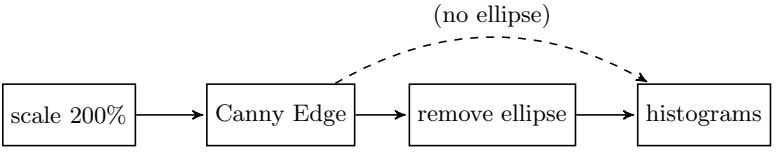


Fig. 2. High-level overview of the descriptor creation

3.1 Database Creation

In order to create the database of reference shapes, we would ideally use actual challenges generated by reCAPTCHA. However, since reCAPTCHA is proprietary, we neither have access to this data nor to the underlying dictionary[2]. To overcome this limitation, we select a reasonably sized dictionary of frequently used words and create our own reference shapes. In order to mimic the real challenges which originate from printed text, our system uses a standard serif font face to render synthetic challenges. Even though this is only a rough and imperfect approximation, its resemblance is sufficient to be covered by the shape context variance. Note that these synthetic challenges are only used for the database; the final performance measurements are derived from real reCAPTCHA challenges.

3.2 Preprocessing

Verbatim challenge images generated by reCAPTCHA contain too much noise and redundancy for shape contexts. This includes JPEG compression artifacts (noise) and the inner area of the stems of characters (redundancy). Therefore we apply a sequence of preprocess steps as depicted in Figure 2. The first step scales the image to 200% of its original size. A subsequent binarization operation then eliminates compression artifacts. Since only the contour of characters is relevant to their shape, we run the Canny edge detector [3] to obtain a contour image. At this point the third generation of reCAPTCHA needs another step to remove the ellipse shaped distortion object which we describe in the next section.

An observant reader may argue that the initial scaling step is technically not necessary since it cannot increase the information available in the image. However, since this is followed by a highly lossy binarization operation, we *reduce* the loss by this measure. Experimental evaluations confirm this by exhibiting higher success rates if the scaling step is performed.

[2] In fact, since the words originate from scanned text where OCR failed, the exact dictionary is not even known to the reCAPTCHA system.

3.3 Ellipse Elimination

Third generation reCAPTCHA challenges (see Figure 1c for an example) contain an ellipse-shaped object under which the colors are inverted. It seems that this object is first drawn as a perfect ellipse and then, along with the challenge word, transformed. Sometimes it is also cut off, apparently because the ellipse extends—or extended prior to the transformation—over the border of the image. Nevertheless, the area still resembles roughly an ellipse. As mentioned in Section 2, precisely this property has been exploited successfully in [7]. We take a slightly different approach here. Instead of trying to directly fit an ellipse onto a set of points, we run a machine learning algorithm that classifies pixels as "ellipse" and "not ellipse." We now describe this mechanism in greater detail.

Ellipse Center Approximation. In order to classify pixels we first require a reference point relative to the ellipse. We use the center of the ellipse for that and present an algorithm to estimate this point. Our algorithm stems from the observation that wherever the ellipse is located, a huge number of black pixels concentrate. It operates as follows. First, repeat the morphological erode operation until only one single connected component of black pixels is left over. Now repeat the dilate operation until the entire image consists of white pixels only. Undo one dilate iteration and finally calculate the center of the remaining black pixels; this is the output of the estimation algorithm. See Figure 3 for an example of the algorithm's operation.

Features. Once the ellipse center has been estimated, a number of features relative to this center p is calculated for every black pixel q_i in the original contour image. These features, arranged in a vector, include amongst others

- the distance and angle from p to q_i,
- the tangent of the edge in q_i,
- pixel density on a line from p to the center point,
- pixel density in the neighborhood of p,

and a number of variations of these features.

Classification Training. Once these features have been calculated, we are interested in learning the mapping that maps each feature vector to its correct class ("ellipse", "not ellipse"). For this we use standard machine learning techniques.

(a) **(b)** **(c)** **(d)**

Fig. 3. Ellipse center estimation. After 7 iterations of erosion only one connected component is left (b). After 58 iterations of dilation only a few pixels close to the center are left over (c). Figure (d) shows these pixels in relation to the original image.

To obtain labeled training data, we classified a set of preprocessed challenges manually by removing the ellipse contour in an image editor. Using OpenCV's boosting algorithm with weak decision tree classifiers on this data then yields a strong classifier.

While this already gives a solid classification result (see Figure 4, left column), there is still room for improvement. For example, each classification decision is made only locally and independently of spatially surrounding classifications. This gives away prior knowledge such as the geometric shape of an ellipse. In order to take this into account, we employ a cascade of classifiers where the ith iteration makes use of knowledge obtained from the $(i-1)$th iteration. Moreover, in each iteration, we calculate a feature that measures the distance to a fitted ellipse for all ellipse-classified pixels.

Accuracy. The fraction of pixels that are classified correctly is denoted by accuracy. We estimate this value with a 10-fold cross validation using 150 weak classifiers and reach a total accuracy of 91.5% after 9 cascade iterations. It takes roughly two hours to train this classifier and less than 300 milliseconds to classify a new example on standard off-the-shelf hardware. Figure 4 presents a classification instance after different iterations of the cascade.

Fig. 4. Cascaded ellipse pixel classification. First row: pixels classified as "not ellipse," second row: pixels classified as "ellipse." From left to right: Classification after iteration 1, 4, and 9.

3.4 Shape Contexts

Once the challenge images are preprocessed, possibly including the ellipse removal step, we are ready to obtain a compact description of the word. As mentioned earlier, our attack uses shape contexts to represent the rendered words.

The key idea of shape contexts is as follows. Let $p_1, \ldots, p_n \in \mathbb{R}^2$ be the points that form the contour line of a shape. For an arbitrary point p_i, called reference point, there are $n-1$ vectors $v_{i,j}$ that describe the location of the other points relative to p_i. Consider now a histogram of the distribution of these vectors $v_{i,j}$ in a polar system[3] centered at p_i. This two dimensional histogram—consisting of angle/distance bins—constitutes a compact but lossy description of the shape

[3] We note that shape contexts are usually associated with log-polar systems (as opposed to polar-linear systems). In our experiments however, we were able to obtain better results with linear distance.

with respect to reference point p_i and is called its shape context. Figure 5 visualizes the histogram bins and the resulting histogram when this transformation is applied to a rendered word. Note that there are n such histograms per shape, one histogram for each contour line point.

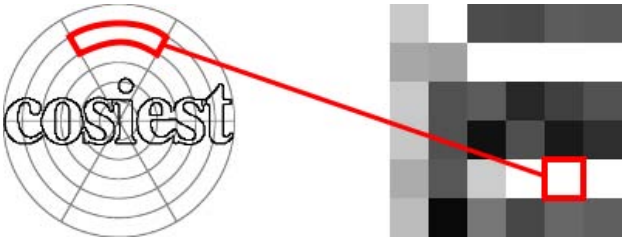


Fig. 5. Histogram bins and the corresponding angle/distance histogram for the center point of the contour line of the word "cosiest"

To measure the similarity between two shapes, one could simply match their corresponding sets of histograms, i.e., find a one-to-one mapping between both sets such that the sum of the distances between each two histograms is minimal with respect to some distance metric. However, it is inefficient and highly redundant to do this on the full set of contour line points. Thus, it makes sense to work with a randomly selected fixed-size subset consisting of, say, 100 histograms. Furthermore, it is not strictly necessary to require a one-to-one mapping between two sets of histograms. Simply selecting the closest match is an acceptable strategy if additional constraints are introduced. One such constraint is the location of the corresponding reference points; requiring a maximal distance here ensures that no points in completely different locations are matched.

In order to further improve the descriptive quality of shape contexts, we use an extended concept called *generalized* shape contexts as proposed in [13] that allows for arbitrary features. Here, Mori et al. additionally record the average tangent of shape points in each histogram bin. This results in a richer description of the shape at the cost of a second set of histograms.

3.5 Efficient Word Matching

We now turn to the online phase of our attack. Given a database of associated shape contexts for each word, our goal is to find the most similar shape for a new query shape. A naive approach comes to mind immediately: compare the query shape with each database shape and output the closest database shape in terms of the distance function. This, however, results in enormous computational cost. Recall that each shape description consists of a set of histograms and that shapes are dictionary words in our case. Matching the histograms of two shapes results in quadratic complexity; a reasonable dictionary size is $20,000$ words. In order to distinguish the many similar words from such a dictionary, the number of reference points/histograms needs to be accordingly high.

To manage this complexity, we propose a search algorithm along the lines of "fast pruning" described in [13]. The general strategy of our algorithm is to start with the full set of database shapes and perform a crude, but fast, comparison against the query shape. Then, the algorithm prunes the most dissimilar shapes from the working set and increases the exactness of the search. Repeated application of this step results in a logarithmic search space reduction. As the shapes become more similar, more time is invested in the comparison. Finally, as soon as the number of shapes in the working set drops below a certain threshold, the algorithm switches to the naive search strategy and outputs the closest match.

The exactness of the search is controlled by the number of reference points used for comparison. For a given number of reference points, the algorithm draws a random subset from all available reference points. A noteworthy consequence is that the algorithm is probabilistic, but this is not so bad because the closest match is not always the correct solution.

Another CAPTCHA-specific pruning strategy which greatly reduces the search space makes use of the fact that the first character and last character are considerably easier to segment. It is immediately clear where the first character starts and the last character ends. A simple and basic approach is to consider an averaged fixed-width section from the start/end of the word. If a character can be detected within this area, a huge portion of the search space is superfluous and thus pruned. In fact, it is already helpful to be able to restrict these key characters to a small set. This is done by employing the shape context matching framework for single characters and selecting the best k matches.

4 Results

Data acquisition. Recall that reCAPTCHA is a proprietary and closed system. This complicates the acquisition of (labeled) challenge/response pairs that are needed for the performance evaluation. One of our methods to collect data is to have humans solve a number of reCAPTCHA challenges and, in the background, record the solution. The advantage of this tedious method is that a human can quickly learn the difference between verification and scan words by close observation. This means that we can deliberately provide a wrong solution for the suspected scan word. If reCAPTCHA confirms this hypothesis by accepting the response, we can be certain that the other word was indeed the verification word. Consequently, we obtain a data set that is not only labeled, but consists also of verification words only, allowing us to derive true performance measurements. In contrast, many reported figures on reCAPTCHA attacks are in fact estimations where the—possibly hidden—underlying assumption is that the attack works equally well on scan and verification words.

Database creation. To build the database of reference words we use a word list prepared by Keith Vertanen [15] which is the intersection of 10 popular word lists. This list contains 22,282 words from the English language. The artificial challenges are then rendered using the Times typeface with negative inter character distance to reflect the overlap situation of real challenges. Shape contexts are created for a 6×6 histogram, i.e., 6 angle bins times 6 distance bins.

Final Results. We stress that our results have been collected from verification words only and thus reflect precisely the success rate of a real attack. See Figure 6 for detailed results. The dictionary success rate in this figure is the (ideal) success rate of our attack *if the challenge word is present* in our dictionary. We are also able to obtain substantially shorter run times in exchange for slightly lower recognition rates.

reCAPTCHA generation	2	3	4
Test set size	496	1005	301
Total success rate	12.7%	5.9%	11.6%
Run time	24.5s	17.5s	15.4s
Dictionary success rate	22%	10.43%	23.5%
First character detected	90.2%	73.2%	84.6%

Fig. 6. Experimental results of our implementation

5 Conclusions

The reCAPTCHA system has been one of the few systems achieving the right balance between usability and security. So far. With its increasing popularity reCAPTCHA has become a major target and the recent attacks reveal significant cracks. Still, because of the centralized system reCAPTCHA allows to switch to a new generation instantaneously. While the concrete attacks may then become ineffective the attack techniques nonetheless improve.

Our attacks for example, achieving a success rate of 5%, show that holistic approaches are feasible, whereas most other attacks are based on segmentation. This is interesting because many systems and techniques so far have been designed to thwart segmentation, e.g., striking out the word. We note that our attacks have not been optimized and thus leave space for improvements. An example is the combination of our holistic approach with partial segmentation, which is—in its current version—only a crude proof of concept of the general technique.

However, we also stress that resistance against automated attacks is not the only concern for CAPTCHAs. Two other dimensions are usability, the ability of humans to solve the CAPTCHA easily, as well as practicality, describing the ability to realize the CAPTCHAs efficiently. For instance, from the security perspective, dictionary-based CAPTCHAs should be used cautiously as they facilitate attacks significantly. It must be said, though, that using dictionaries supports humans in recognizing words. Another worthwhile point is that reCAPTCHA is based on the idea that solving a CAPTCHA helps digitizing books. This idea may incite users to solve such otherwise unpopular puzzles, thus improving the overall acceptance of CAPTCHAs.

Overall, the recent attacks on reCAPTCHA somehow leave us in a vague state. It remains an open problem if there exist CAPTCHAs which are simultaneously secure, usable, and practical. Given the status of CAPTCHAs in modern login services, a CAPTCHA system meeting all these requirements is of great demand.

On a slightly positive note, however, even though our results indicate that the security of yet another CAPTCHA system has become dubious, there is also an upside in the particular case of reCAPTCHA. By design, any system that breaks reCAPTCHA is a step towards better OCR software.[4] Our results indicate that Shape Contexts could be a valuable fallback solution in the domain of character recognition.

Acknowledgements

We thank the anonymous reviewers for valuable comments. Paul Baecher and Marc Fischlin are supported by the Emmy Noether Grant Fi 940/2-1 of the German Research Foundation (DFG).

References

1. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: reCAPTCHA: Human-based character recognition via web security measures. Science 321(5895), 1465–1468 (2008) Cited on page 1
2. Belongie, S., Malik, J., Puzicha, J.: Shape context: A new descriptor for shape matching and object recognition. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) NIPS, pp. 831–837. MIT Press, Cambridge (2000) Cited on pages 2 and 4
3. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 8, 679–698 (1986), http://portal.acm.org/citation.cfm?id=11274.11275 Cited on page 6
4. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Building segmentation based human-friendly human interaction proofs (HIPs). In: Baird, H.S., Lopresti, D.P. (eds.) HIP 2005. LNCS, vol. 3517, pp. 1–26. Springer, Heidelberg (2005) Cited on page 4
5. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In: CEAS (2005) Cited on page 4
6. Govindaraju, V., Krishnamurthy, R.K.: Holistic handwritten word recognition using temporal features derived from off-line images. Pattern Recognition Letters 17(5), 537–540 (1996) Cited on page 5
7. Houck, C.W.: Decoding recaptcha (2010), http://www.n3on.org/projects/reCAPTCHA/docs/reCAPTCHA.docx Cited on pages 3 and 6
8. Lavrenko, V., Rath, T.M., Manmatha, R.: Holistic word recognition for handwritten historical documents. In: DIAL, pp. 278–287. IEEE Computer Society Press, Los Alamitos (2004) Cited on page 5
9. Lladós, J., Roy, P.P., Rodríguez, J.A., Sánchez, G.: Word spotting in archive documents using shape contexts. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) IbPRIA 2007. LNCS, vol. 4478, pp. 290–297. Springer, Heidelberg (2007) Cited on page 4

[4] It must be said, though, that a successful attack may only achieve a recognition rate of say, 10% of the challenges, which is too low for a full-fledged OCR program.

10. Madhvanath, S., Govindaraju, V.: Contour-based image preprocessing for holistic handwritten word recognition. In: ICDAR, pp. 536–539. IEEE Computer Society Press, Los Alamitos (1997) Cited on page 5
11. Madhvanath, S., Govindaraju, V.: The role of holistic paradigms in handwritten word recognition. IEEE Trans. Pattern Anal. Mach. Intell. 23(2), 149–164 (2001) Cited on page 5
12. Mori, G., Belongie, S., Malik, J.: Shape contexts enable efficient retrieval of similar shapes. In: CVPR, vol. 1, pp. 723–730. IEEE Computer Society Press, Los Alamitos (2001) Cited on page 4
13. Mori, G., Belongie, S.J., Malik, J.: Efficient shape matching using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. 27(11), 1832–1837 (2005) Cited on page 9
14. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: CVPR, vol. 1, pp. 134–144. IEEE Computer Society Press, Los Alamitos (2003) Cited on page 4
15. Vertanen, K.: Words in 10 lists (2010), http://www.keithv.com/software/ Cited on page 10
16. Wilkins, J.: Strong CAPTCHA guidelines v1.2 (2009), http://www.bitland.net/ Cited on page 3

From Multiple Credentials to Browser-Based Single Sign-On: Are We More Secure?*

Alessandro Armando[1,2], Roberto Carbone[2], Luca Compagna[3], Jorge Cuellar[4], Giancarlo Pellegrino[3], and Alessandro Sorniotti[5]

[1] DIST, Università degli Studi di Genova, Italy
[2] Security & Trust Unit, FBK, Trento, Italy
[3] SAP Research, Mougins, France
[4] Siemens AG, Munich, Germany
[5] IBM Research Zurich, Rüschlikon, Switzerland

Abstract. Browser-based Single Sign-On (SSO) is replacing conventional solutions based on multiple, domain-specific credentials by offering an improved user experience: clients log on to their company system once and are then able to access all services offered by the company's partners. By focusing on the emerging SAML standard, in this paper we show that the prototypical browser-based SSO use case suffers from an authentication flaw that allows a malicious service provider to hijack a client authentication attempt and force the latter to access a resource without its consent or intention. This may have serious consequences, as evidenced by a Cross-Site Scripting attack that we have identified in the SAML-based SSO for Google Apps: the attack allowed a malicious web server to impersonate a user on any Google application. We also describe solutions that can be used to mitigate and even solve the problem.

1 Introduction

To provide access to restricted services, web applications assign digital credentials to registered users and require users to prove possessions of these credentials to receive access to protected resources. As web applications become more and more widespread, users must handle an increasing number of authentication credentials to establish security contexts with web applications. This is not only an annoying aspect of the current state of affairs, but has serious implications on the security of these systems as users tend to use weak passwords and/or to reuse the same password on different web applications.

Browser-based SSO solutions aim at improving this state of affairs by allowing users to log in once and by giving them subsequent access to multiple web applications. At the core of a browser-based SSO solution lies a browser-based

* This work has partially been supported by the FP7-ICT Projects AVANTSSAR (no. 216471) and SPACIOS (no. 257876), and by the project SIAM funded in the context of the FP7 EU "Team 2009 - Incoming" COFUND action. Furthermore the authors would like to thank Brian Eaton, Scott Cantor, Matteo Grasso, and the SAP NetWeaver SIM team for the valuable discussions and feedback they provided.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 68–79, 2011.

authentication protocol. Three roles take part in the protocol: a client (C), an identity provider (IdP) and a service provider (SP). The objective of C, typically a web browser guided by a user, is to get access to a service or a resource provided by SP. IdP authenticates C and issues corresponding authentication assertions. Finally, SP uses the assertions generated by IdP to decide on C's entitlement to the requested resource.

A number of solutions for browser-based SSO have been put forward, e.g. Microsoft® Passport [11], the Liberty Alliance project [12], the Shibboleth Initiative [9], and OpenId [15]. The OASIS *Security Assertion Markup Language* (SAML) 2.0 Web Browser SSO Profile (SAML SSO, for short) [13] is an emerging standard in this context: it defines an XML-based format for encoding security assertions as well as a number of protocols and bindings that prescribe how assertions must be exchanged in a variety of applications and/or deployment scenarios. Prominent software companies base their SSO implementations on SAML SSO. For example, Google has developed a SAML-based SSO service for its popular web applications (namely Gmail, Google Calendar, Talk, Docs and Sites), called the SAML-based SSO for Google Apps [5].

The security of SAML SSO critically relies on a number of assumptions on the trustworthiness of the principals involved as well as on the security of the transport protocols used to exchange messages. In this paper we argue that one of the assumptions on the security of the transport layer (i.e., that communication between the client and the service provider must be carried over a unilateral SSL 3.0 or TLS 1.0 connection) can only be met in practice in a way that leaves the protocol vulnerable to an authentication flaw. We discuss how this flaw can be exploited in general as well as on a number of prominent SAML-based SSO solutions, including the SAML-based SSO for Google Apps that is used by over one million business customers. Our findings show that the authentication flaw can be seriously exploited in actual deployments of SAML SSO. For instance, a severe attack could be mounted on the SAML-based SSO for Google Apps in which a malicious web server could impersonate the victim user on any Google application. In the paper we also provide solutions that allow the authentication flaw and its exploitations to be mitigated or even eliminated.

To the best of our knowledge neither the authentication flaw on the SAML SSO nor the vulnerability of the SAML-based SSO for Google Apps reported in this paper are publicly known. We are currently informing US-CERT and the relevant vendors about our findings. In response to our vulnerability report Google has already patched their implementation of their SAML SSO solution.

What about the original question in the title, are we more secure with SAML SSO than with multiple credentials? This question does not have a trivial answer and certainly a positive answer cannot be given as long as there are unaddressed issues such as the vulnerability we present in this paper. In addition, since the security considerations brought forward by this paper do not apply to SAML SSO only, we believe that other browser-based SSO protocols may suffer from similar vulnerabilities. We are currently extending our analysis to other SSO solutions to ascertain this.

Structure of the paper. In the next section we introduce the SAML SSO profile for web-based authentication. In Section 3 we present the authentication flaw on the SAML SSO, and in Section 4 describe how it can be exploited on actual implementations. In Section 5 we provide a number of solutions for the flaw. Last but not least, in Sections 6 and 7 we discuss some of the related work and present conclusions.

2 The SAML 2.0 Web Browser SSO Profile

SAML SSO provides a standardized, open, interoperable SSO solution applicable in a multitude of environments and situations, and can therefore be instantiated according to the specific requirements posed by the application scenario. In this paper we focus on one of its most widely used instantiations, the SP-Initiated SSO with Redirect/POST Bindings, whose typical use case is described in [14]. In the remainder of this paper we will refer to this use case as *the SAML SSO use case* and to the associated protocol as *the SAML SSO Protocol*.

In Figure 1 we capture the most important steps of the *SAML SSO Protocol*, abstracting away the steps that are irrelevant for our analysis, such as—among others—the IdP discovery phase. In step S1, C asks SP to provide the resource located at URI, say *Resource*(URI), without having a valid, active logon session (i.e. security context) with SP. SP then initiates the *SAML Authentication Protocol* by sending to C an HTTP redirect response (status code 302) to IdP, containing an authentication request AuthReq(ID,SP), where ID is a (pseudo-)randomly generated string uniquely identifying the request (steps A1 and A2). A frequent implementation choice is to use the `RelayState` field to carry the original URI that the client has requested (see [14]).

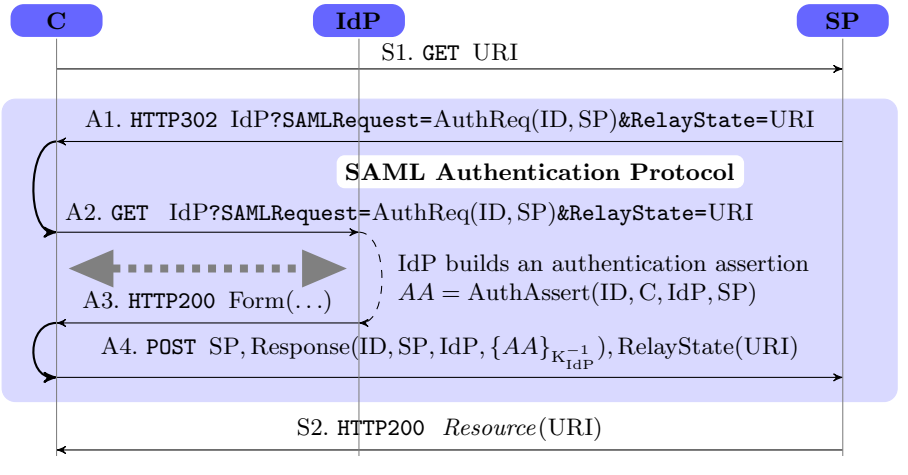


Fig. 1. SAML SSO Protocol: SP-Initiated SSO with Redirect/POST Bindings

If C does not have an existing security context with the IdP, then IdP challenges C to provide valid credentials. If the authentication succeeds, the IdP creates the local security context, builds an authentication assertion as the tuple $AA = \text{AuthAssert}(\text{ID}, \text{C}, \text{IdP}, \text{SP})$, and places it in a response message $Resp = \text{Response}(\text{ID}, \text{SP}, \text{IdP}, \{AA\}_{\text{K}_{\text{IdP}}^{-1}})$, where $\{AA\}_{\text{K}_{\text{IdP}}^{-1}}$ is the assertion signed with $\text{K}_{\text{IdP}}^{-1}$, IdP's private key. IdP then places *Resp* and the value of `RelayState` received from the SP into an HTML form (indicated as Form(...) in Figure 1) and sends the result back to C in an HTTP response (step A3) together with some script that automatically posts the form to the SP (step A4). This completes the SAML Authentication Protocol. SP can then deliver the requested resource, *Resource*(URI), to C (step S2), and the SAML SSO Protocol completes as well.

Note that the steps at message S1 and S2 admittedly fall outside of the scope of the standard, and their implementation is left free. In this paper we capture steps S1 and S2 as described in *the SAML SSO use case*; a number of commercial SAML SSO solutions indeed adopt similar approaches to implement those steps.

As pointed out in [2] the security of the protocol critically relies on (unstated) assumptions about the trustworthiness of the participants involved and about the transport protocols used to exchange the protocol messages; we shall review these in the next Sections.

2.1 Trust and Transport Protocol Assumptions

The above protocols work under the assumption that *(i)* IdP is not compromised, i.e. it is not under the control of an intruder and it abides by the rules of the protocol and *(ii)* IdP is trusted by SP to generate authentication assertions about C. Even if they are not explicitly stated in the SAML 2.0 specifications, these are very reasonable assumptions to make and, in fact, both protocols are useless if the IdP is not trusted to generate authentication assertions about C or if there is the doubt that the IdP is compromised. However, *we do not assume that all SPs which C may play the protocol with are uncompromised*. In other words, unlike [8], we want to consider also those situations in which C runs the protocol with compromised SPs in order to determine whether they affect the security of sessions of the protocol played with other uncompromised SPs. This is very important as SPs are usually managed by different organizations that do not always share trust relationships.

The SAML 2.0 specifications repeatedly state the following assumptions of the transport protocols used to carry the protocol messages:

(TP1) Communication between C and SP is carried over a unilateral SSL 3.0 or TLS 1.0 channel (henceforth called SSL), established through the exchange of a valid certificate (from SP to C).

(TP2) Communication between C and IdP is carried over a unilateral SSL channel that becomes bilateral once C authenticates itself on IdP. This is established through the exchange of a valid certificate (from IdP to C) and of valid credentials (from C to IdP).

2.2 Security Requirements

The SAML specifications do not explicitly state the security properties that the SAML SSO Protocol and the SAML Authentication Protocol are expected to achieve. By comparison with classic web authentication schemes, it is however natural to expect that at the end of the *SAML SSO Protocol*, the following security property is fulfilled:

(P1) SP and C mutually authenticate and agree on the value URI

As pointed out in [10], different definitions of authentication are possible. The notion of authentication we consider in this paper includes *recentness*, i.e. the fact that the principal being authenticated recently took part in the protocol run so as to exclude replay attacks.

We note that the SAML Authentication Protocol, the building block of the SAML SSO Protocol, is only able to guarantee the property

(P2) SP authenticates C

The converse is not true, i.e., the SAML Authentication Protocol does not provide to C any guarantee on SP's identity; indeed in message A1, SP may instruct IdP to force C to redirect message A4 to an arbitrary location. Even the use of SSL certificates only guarantees that there is no man-in-the-middle in the communications between C and the recipient of message A4.

In the remainder of this paper, we will investigate whether the SAML SSO Protocol, constructed with a building block that only guarantees **(P2)**, is able to fulfill the original property **(P1)**, and we will show that the fulfillment of this property is not automatically guaranteed; in particular depending on the implementation choices, a malicious SP may be able to hijack C's authentication attempt and force the latter to access a resource without its consent or intention.

3 An Authentication Flaw in the SAML SSO Protocol

An analysis of the SAML specifications reveals that the standard does not specify whether the messages exchanged at steps S1 and A4 must be transported over the same SSL channel or whether two different SSL channels can be used for this purpose. In other words, there is a certain degree of ambiguity on how assumption (TP1) of Section 2 can be interpreted.

The reuse of the SSL channel established at step S1 to also transport the message at step A4 is at first sight the most natural option. However this is difficult to achieve in practice for a number of reasons:

Resuming SSL sessions. The use of a single SSL session for the exchange of different messages cannot be guaranteed as, e.g., the underlying TCP connection might be terminated (e.g. timeout, explicitly by one of the end points), an SSL server could not resume a previously established session, or a client might be using a browser that very frequently renegotiates its SSL session.[1]

[1] See, for instance, `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame2.doc_5.1/am51_webseal_guide54.htm`

Software modularity. Nowadays, software is designed to be increasingly modularized, capitalizing on layering and separation of concerns. This may result in the fact that—within SP implementations—the software module that handles SAML messages has no access to the internal information of the transport module that handles SSL. Thus, the information on whether the client has used a single SSL session or two different ones may not be available.

Distributed SPs. The SAML SP may be distributed over multiple machines, for instance, for work-balancing reasons. This results in physically different SSL endpoints, with the inherent impossibility of enforcing a single session for all communications between SP and C.

We have extended the formal model discussed in [2] to faithfully capture the SAML SSO use case in which the messages of steps S1 and A4 can be transported over different SSL sessions and fed it to a state-of-the-art model-checker for security protocols [1]. (See Section 6 for more details.) The model checker detected the attack depicted in Figure 2, thereby witnessing a violation of property **(P1)** in the *SAML SSO Protocol*.

The attack involves four principals: a client (c), an honest IdP (idp), an honest SP (sp) and a malicious service provider (i). The attack is carried out as follows: c initiates the protocol by requesting a resource uri_i at SP i. Now i, pretending to be c, requests a different resource uri at sp and sp reacts according to the standard by generating an Authentication Request, which is then returned to i. Now i maliciously replies to c by sending an HTTP redirect response to idp containing AuthReq(id, sp) and uri (instead of $\text{AuthReq}(\text{id}_\text{i}, \text{i})$, and uri_i as the standard would mandate). The remaining steps proceed according to the standard. The attack makes c consume a resource from sp, while c originally asked for a resource from i.

Note that the attack is possible essentially because the client—usually a normal browser with no knowledge of the SAML protocol—has no means of verifying whether the authentication request and the authentication assertion are related to the initial request.

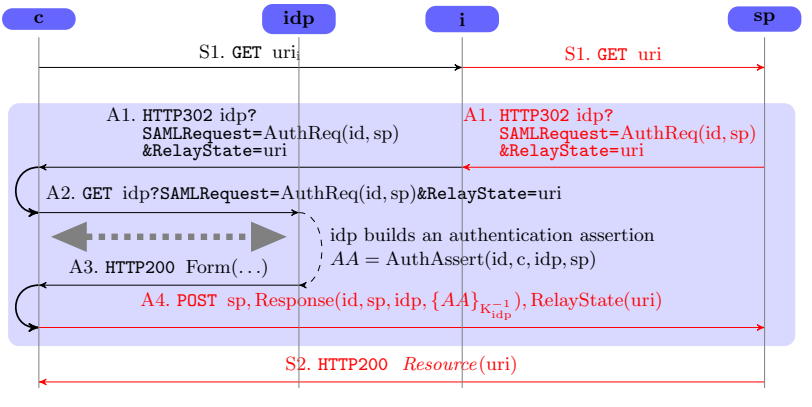


Fig. 2. Authentication Flaw of the SAML 2.0 Web Browser SSO Profile

Interestingly enough, standard username/password authentication mechanisms do not suffer from this authentication flaw. To see this, let us assume that C has no active sessions with service providers SP1 and SP2; let us also assume that C's usernames and passwords are different for each SP.[2] Then under no circumstance can SP1 hijack C's authentication attempt and unawarely and automatically force it to consume a protected resource at SP2. From this point of view, the advantage of domain-specific credentials in the control of the user is that the user knows exactly for whom the credentials are intended upon providing them. With SSO, "binding" the views of the user and of the service provider is not so easy.

Note that the attack would be prevented if sp could enforce that the initial request and authentication response are carried over the same secure channel, but we have previously explained why this requirement is very difficult to achieve in practice. Note also that requiring digitally signed authentication requests would not fix the vulnerability; indeed the authentication request is actually generated by the honest service provider, and only blindly forwarded to the client by the attacker; the signature is therefore valid and will be accepted.

Even more interestingly, the attack does not strictly require a malicious service provider in order to be successful. Any malicious web server i would be able, upon a request from c, to mount the attack provided that *(i)* c is a client of sp and *(ii)* c has an active authentication context with idp.

The attack in Figure 2 can be exploited in a number of ways:

Delivery of an unrequested resource. The most trivial exploitation of the flaw consists in the attacker forcing the client to receive a different protected resource from the initially requested one. The same exploitation may also be mounted if a malicious web server redirects the browser to a legitimate SP before the SAML SSO Protocol starts. However this attack can be prevented by using well-known browser-side plugins that restrict HTTP redirections (e.g., the NoRedirect addon for Firefox). By allowing only IdP-to-SP and SP-to-IdP redirections, the delivery of an unrequested resource upon redirection outside of the SAML SSO Protocol is prevented, but a malicious SP can still mount the one depicted in the Figure 2.

Launching pad for cross-Site Request Forgery (XSRF) attacks. This attack assumes that the URI that was initially requested did not point to a resource, but rather contained a URL-encoded command, such as a request for the change of some settings or user's preferences, for the deletion of some resource or for the annulment of/committing to an action, such as the purchase of a paid good. Depending on the output provided by the execution of the command, the client may or may not be able to detect the attack. This type of attack is even more pernicious than classic XSRF, because XSRF requires C to have an active session with SP, whereas in this case, the session is created automatically hijacking C's authentication attempt.

[2] If this assumption does not hold, C is vulnerable to a number of other trivial attacks anyway.

Launching pad for cross-Site Scripting (XSS) attacks. It is straightforward to see that this attack also constitutes a launching pad to reflected XSS attacks, i.e. XSS attacks that can be triggered by visiting a maliciously-crafted URL. In addition, a vanilla implementation of the SAML SSO protocol exposes the `RelayState` field to a possible injection of malicious code that may be executed at the honest SP side. Although the SAML standard recommends to protect the integrity of this field, our experience shows that this often is not the case (see Section 4). In addition, unlike normal XSS attacks, where the attacker has to rely on social engineering (phishing, spam and so forth) to lure a victim into clicking on a malicious link, an exploitation of the vulnerability paves the way for systematically luring victims into visiting URIs that may be vulnerable to cross-site scripting attacks. Note also that in this case, unlike in the previous exploitations, the client is not suspicious about receiving a different resource than the one requested. On the contrary, because arbitrary code can be embedded in uri, a redirection to uri_i, the page that c initially requested, can be eventually forced at the end of the attack. As an example, if uri is forged as `javascript:window.open('`uri_i`'+document.cookie)` the client would be victim of the theft of its cookies for the domain sp through a visit to the requested uri_i.

Although in this paper we focus on the SP-initiated SSO protocol, it is worth mentioning that IdP-Initiated flows may suffer from *login CSRF* attacks [3], whereby the attacker forges a cross-site request to the login form and, logs the victim into a honest web site *as the attacker*.

4 Exploitations in Actual Deployments

An interesting question that we also address in this paper is whether exploitations of the abstract weakness of the standard are possible in actual deployments of the SAML SSO Protocol. To this end, we have analyzed various SAML-based SSO solutions available on the market, including SAML-based SSO for Google Apps, SimpleSAMLphp as deployed for Foodle (see `https://foodl.org`), and a deployment of the Novell Access Manager 3.1 in a real industrial environment. All these deployments support the SAML SSO use case; not surprisingly, by inspecting SSL messages we verified that the SPs employed in these deployments accept and process a SAML response flowing on a different SSL channel than the one used to deliver the SAML request.

Our analysis of the SAML-based SSO for Google Apps shows that by exploiting the weakness of the standard, a malicious SP can force C to consume a resource from Google, for instance, visiting any page of the gmail service. mailbox. This trivial attack is however easily detected by C, and does not bring any real advantage to the attacker. Definitely more serious for the over one million business customers of Google Apps was the XSS attack we were able to execute and that allowed the malicious SP to steal the C' cookies for the Google domain and thus to impersonate C on any Google application. The abstract flaw of Figure 2 served indeed as launching pad for this XSS: because of missing

sanitization, an attacker could inject malicious code into the `RelayState` field and have it successfully executed on the client's browser as if coming from the Google domain (thus circumventing the same origin policy). In other words, the combination of the abstract flaw and the missing sanitization was the key to mount the XSS attack. The past tense is in order here since, as soon as we found this attack, we informed Google, who promptly patched the issue.

We have been able to mount a similar XSS attack on the SAML SSO solution of the Novell Access Manager 3.1 as deployed in a real industrial environment. In this deployment `RelayState` is not used to store the URI;instead, a URL-encoded parameter is used to this end, and this parameter is not sanitized.

The SimpleSAMLphp, as deployed in Foodle, stores the initially requested URI into the URL parameter `ReturnTo`. Although that field is not sanitized, we have not been able to mount any XSS. The reason is that SPs running SimpleSAMLphp additionally use cookies that block the abstract flaw we discovered. We will detail this solution in the next section.

The findings presented above show that the authentication flaw we discovered can be exploited on actual deployments of the SAML SSO Protocol, even leading to major security issues. We have informed Novell and UNINETT (the developer and maintainer of SimpleSAMLphp) about these findings as well as the US-CERT so that other vendors implementing and deploying SAML-based SSO solutions can get advantage of this information.

5 Fixing the Vulnerability

The root of the problem of the authentication flaw presented earlier lies in the following two main factors:

1. Clients are not able to link the Authentication Request they receive from the SP in step A1 with their initial requests for a resource issued in step S1;
2. The SP is not able to enforce that the messages exchanged with C (cf. steps S1, S2, A1, and A4) are carried over the same channel.

We have verified that—could one of the two causes be removed—the vulnerability would no longer be exploitable. We emphasize nonetheless that the *SAML SSO Protocol* alone neither achieves property **(P1)** nor mandates the implementation of any of these solutions, thus leaving a vanilla implementation in principle flawed.

The challenge is to fix the vulnerability with minimal changes so that existing solutions can be secured without radical modifications to the software components (e.g. SAML ECP profile) or to the standard. In what follows, we outline a number of possible solutions, highlighting their strengths and shortcomings.

Cookies. A standard way of enforcing bindings on sessions is implemented using session cookies. With reference to Figure 1, by setting a session cookie in step A1 and expecting to receive it back on message A4, SP could check that the communication has occurred with the same client. However, cookies only

provide means to mitigate the problem—albeit sufficient in many scenarios—and do not represent a complete countermeasure. Indeed cookies are designed to be difficult to steal and it is not as hard to set them. For instance, cookies with the "Secure" flag on (which instructs the browser not to transmit them over unencrypted channels) can be set over unprotected connections. (The latest versions of IE and of Firefox allow this.) In practice an attacker could circumvent the protection offered by cookies by *(i)* setting a cookie for the victim SP through injected Javascript or HTML META tags; *(ii)* corrupting the proxy discovery phase setting up a rogue wpad or dhcp server, thus becoming the user's proxy; *(iii)* performing ARP poisoning thus becoming the victim's default gateway.

Feedback from the user. As seen in the preceding Sections, the user may initiate the SAML SSO profile, authenticating to an SP without actually having explicited requested anything from it. This can be avoided if the IdP informs the user about the attempt to access URI on the SP during the authentication and asks for an explicit consent before issuing the authentication assertion to SP. In this way, the user may realise that the authentication is going to be sent to a different SP than expected and may be given the possibility to stop the protocol. This solution has a number of drawbacks: first of all, it forces a security decision upon a (possibly technically unaware) user, who is asked to tell apart legitimate SP-to-SP redirections from malicious ones. In addition, it breaks the seamlessness of SSO, in which the authentication process is supposed to be carried out with minimal interactions with the user.

Self-signed client certificates. A simple, yet effective way to ensure SP that it is interacting with the same client is to provide the latter with a self-signed certificate. The solution goes as follows: during the first SSL session (cf. steps A1 and A2 in Figure 1) C is asked to present the certificate. SP will then generate an Authentication Request and its ID field is set to $n \,||\, \texttt{HMAC}_{K\,||\,n}(\texttt{RSA modulus})$, where n is a nonce, K is a secret known only to SP, `RSA modulus` is the RSA modulus of the public key contained in the client's certificate. HMAC is the well-known HMAC keyed hash function [4] and $||$ denotes the concatenation. After this, SP deletes all state information and sends the Authentication Request to C. During the second SSL session (cf. steps A3 and A4), C is again asked for the certificate and the same certificate will be delivered to SP. The standard requires the *InResponseTo* field of the Response message to contain the same value of the ID field of the Authentication Request message: therefore SP can parse such field as n' and H' and then check whether $H' = \texttt{HMAC}_{K\,||\,n'}(\texttt{RSA modulus})$.

Note that *(i)* the client can easily self-generate a certificate; alternatively, the SP can offer the client to forge one on his behalf; *(ii)* the certificate is *not* expected to carry information about the identity of the user; in particular, it is not used to assess the user identity; and *(iii)* during the SSL handshake, the browser proves knowledge of the private key; the approach therefore guarantees with overwhelming probability that a malicious third party cannot forge a copy of the same certificate since – in case of certificates that use RSA encryption for instance – it would entail breaking the RSA hardness assumption.

6 Related Work

Pfitzmann et al. [16,17,7] lay the theoretical basis for a rigorous analysis of web-based federated identity-management protocols (e.g. the SSO protocol proposed by Liberty Alliance in 2002). They discuss some security vulnerabilities and possible preventive measures. Some of these results have been fed into the Liberty Alliance project and indirectly into the SAML 2.0 standard.

Security analyses of the SAML SSO v1.0 are presented in [6] and in [8]. The security analysis presented in our paper refers to SAML SSO v2.0, the latest version of the standard. Moreover, in our work we focus on scenarios that are most likely to occur in actual deployments. For instance, unlike [8] we do not assume that SPs are trustworthy and unlike [6] we assume that messages are exchanged over secure channels as recommended by the standard.

In [2] we provide a formal model of the SAML SSO protocol as well as of a variant implemented in the SAML-based SSO for Google Apps. By using a model checker, we discovered a subtle man-in-the-middle attack on the SAML-based SSO for Google Apps. In reaction to this discovery Google has modified the implementation of the protocol. The version of the protocol used by the SAML-based SSO for Google Apps we described in Section 4 is the one currently in use by Google and therefore does not suffer from the attack reported in [2]. Interestingly, in [2] we did not find any attack on the Web Browser SAML 2.0 SSO profile as in our analysis we assumed that communication between C and SP is carried over a single unilateral SSL channel. We have adapted that formal model so to allow the messages of steps S1 and A4 to be transported over different SSL sessions and used the SATMC model-checker to analyze this new specification. This has allowed us to discover the previously unknown attack described in Section 3.

7 Conclusions

Authentication protocols are notoriously difficult to get right, even more so for browser-based authentication protocols because "browsers, unlike normal protocol principals, cannot be assumed to do nothing but execute the given security protocol" [7]. In this paper we have showed that browser-based SSO protocols are no exception. We have presented an authentication flaw in the SAML SSO, discussed how this flaw can be generally exploited, and reported related security issues that we have detected in actual SAML-based SSO solutions developed by prominent software companies, including a severe attack on the SAML-based SSO for Google Apps. We have finally presented a number of possible solutions that mitigate or even solve the problem. As a part of our future work we plan to extend our analysis to other SSO solutions.

References

1. Armando, A., Carbone, R., Compagna, L.: LTL Model Checking for Security Protocols. Journal of Applied Non-Classical Logics, special issue on Logic and Information Security, 403–429 (2009)

2. Armando, A., Carbone, R., Compagna, L., Cuéllar, J., Tobarra, M.L.: Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In: FMSE. ACM, New York (2008)
3. Barth, A., Jackson, C., Mitchell, J.C.: Robust defenses for cross-site request forgery. In: 15th ACM Conference on Computer and Communications Security (CCS 2008) (2008)
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
5. Google. Web-based SAML-based SSO for Google Apps (2008), http://code.google.com/apis/apps/sso/saml_reference_implementation_web.html
6. Groß, T.: Security analysis of the SAML Single Sign-on Browser/Artifact profile. In: Proc. 19th Annual Computer Security Applications Conference. IEEE, Los Alamitos (December 2003)
7. Groß, T., Pfitzmann, B., Sadeghi, A.-R.: Browser model for security analysis of browser-based protocols. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 489–508. Springer, Heidelberg (2005)
8. Hansen, S.M., Skriver, J., Nielson, H.R.: Using static analysis to validate the SAML single sign-on protocol. In: WITS 2005. ACM Press, New York (2005)
9. Internet2. Shibboleth Project (2007), http://shibboleth.internet2.edu/
10. Lowe, G.: A hierarchy of authentication specifications. In: Proc. CSFW. IEEE, Los Alamitos (1997)
11. Microsoft. Windows Live ID, https://www.passport.net/
12. OASIS. Identity Federation. Liberty Alliance Project (2004), http://www.projectliberty.org/resources/specifications.php
13. OASIS. SAML V2.0 (April 2005), http://docs.oasis-open.org/security/saml/v2.0/
14. OASIS. SAML V2.0 – Technical Overview (March 2007), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
15. OpenID Foundation. OpenID Specifications (2007), http://openid.net/developers/specs/
16. Pfitzmann, B., Waidner, M.: Analysis of Liberty Single-Sign-on with Enabled Clients. IEEE Internet Computing 7(6) (2003)
17. Pfitzmann, B., Waidner, M.: Federated identity-management protocols. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2003. LNCS, vol. 3364, pp. 153–174. Springer, Heidelberg (2005)

Quantifying the Effect of Graphical Password Guidelines for Better Security

Mohd Jali[1,3], Steven Furnell[1,2], and Paul Dowland[1]

[1] Centre for Security, Communications and Network Research (CSCAN),
Room A304, Portland Square, University of Plymouth, Plymouth PL4 8AA, UK
[2] School of Computer & Security Science, Edith Cowan University,
Perth, Western Australia
[3] Faculty of Science & Technology, Universiti Sains Islam Malaysia,
Nilai, 71800, Negeri Sembilan, Malaysia
zalisham@usim.edu.my

Abstract. Authentication using images or graphical passwords is one of the possible alternatives for traditional authentication based upon passwords. This study aims to investigate the practicality of giving guidelines or advice to users before they start choosing their image passwords, the effectiveness of using a smaller tolerance (clickable areas) and the optimum combination of click and image passwords. An alternative graphical prototype known as the Enhanced Graphical Authentication Scheme (EGAS) was developed in order to achieve these aims which implemented two different types of data collection (internal and external). From the findings, both internal and external groups indicated that the implementation of guidelines alone cannot guarantee the security of image passwords created by participants; but, in combination with other usability measurements this study has shown positive outcomes.

Keywords: Graphical passwords, Authentication, Usability, Security, HCI.

1 Motivation

Using images to authenticate users is one possible alternative for password-based authentication. Previous work has divided image-based authentication into three categories; namely 'click-based', 'choice-based' and 'draw-based'. The click-based approach refers to the action of clicking on the provided/chosen image(s) (i.e. selecting an element of the image), choice-based refers to the action of selecting a series of images (i.e. choosing images from a selection on screen) and draw-based refers to the action of drawing/sketching in order to be authenticated.

Regardless of the methodologies, previous studies have reported positive results, especially in the aspects of recall and memorability (i.e. participants were able to remember their secrets (i.e. image passwords) accurately after long periods of time) and usability (i.e. using images is user friendly) [1], [2] and [3]. Conversely, studies have also reported the disadvantages. Davis et al. [7] and Tullis and Tedesco [8] found that users chosen secrets were influenced by gender. Chiasson et al. [4] reported that the concept of clicking on images (e.g. Passpoint [5]) was not secure as

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 80–91, 2011.

users tended to create hotspots (i.e. focussing upon one area in an image) and generating similar patterns (e.g. a straight line from top-bottom or left-right). Oorchot et al. [6] claimed that it was possible to crack users' secrets regardless of the background image, with the study by Everitt et al. [9] reporting that having multiple secrets resulted in more errors when compared with password-based authentication.

With respect to security, the main problem with the click-based method can be referred to as 'hotspot' while the problem with the choice-based method can be referred to as 'hot-image'. The problem of hot-image happens when a similar image is selected by many users. This problem could also be associated when users choose similar categories/themes or through gender preferences (e.g. males choose cars and females choose flowers). The hotspot problem could occur in two conditions. Firstly, the user clicks within the same or similar point on the given image or clicks on the same point or area when two or more images are given. Secondly the user produces predictable shapes such as straight lines and clicks on obvious/predictable objects within the image. Studies related to security in graphical passwords can be found in [10], [11], [12] and [13].

In an attempt to address or reduce the aforementioned problems and at the same time maintain users' memorability, many studies have been published with regards to the effect of using various types of images. Examples include using images of cartoon characters [3], images of geometric shapes [14] and using images that were later transformed into unclear or distorted forms during login [15] and [16]. With respect to the click-based method, a technique known as persuasion has been proposed [17] where the software recommends to the user possible 'safe' areas in which to create their secrets.

As far as the authors are aware, no study was found to have investigated or introduce user guidelines as part of the enrolment process. Therefore, the authors introduced a set of guidelines for graphical authentication, referred to as the Graphical Password Guidelines (GPG) which was presented to the user before they began choosing their secrets.

The authors also conjectured that GPGs on their own (Table 1) would not be a universal solution due to inherent human behaviour (i.e. certain users, although aware of the guidelines, sometimes violate the rules). To address this, restrictions were applied during registration. Two restrictions implemented in this study are as follows:

1. Users were only permitted to choose one image per category.
2. Users were not permitted to click on the same areas within an image. If they choose more images, they were also not permitted to click on the same area within the images.

The above restrictions together with the GPG were integrated into a software prototype. The software applied these restrictions by displaying warning messages if the software identified the user attempting to breach the rules.

The study was conducted in order to examine the impact on usability as well as user perception towards the introduction of the GPGs and image selection restrictions. Each participant had two types of secret; namely click-secrets (based upon the action of clicking on an image) and image-secrets (based upon the action of choosing a sequence of images). In addition to this, the study aimed to find ideal (usable and secure) combinations of click and image-secrets. A third investigation was undertaken

to evaluate the impact of reducing the tolerance of the click positions. Tolerance can be explained as the extent of the area surrounding the users' secret clicks which are still accepted as legitimate. Prior research has indicated that participants were quite good when entering their secrets, both during registration and login [19]. Thus, the authors believed that using a smaller tolerance is possible and for this reason, users' performance when using smaller tolerance was investigated.

Table 1. Graphical password guidelines

Task	Guideline	Explanation
Choosing images	Choose different themes and images	Users perceives image differently and previous studies have found gender bias in user image selections [7], [8] and [18]. As a result, the user is advised to choose different images, the image itself should not related to gender and more importantly, they are advised to choose images that they think could offer them memorable areas for placing their secret clicks.
	Try to avoid imagery that could be associated with your gender	
	Please choose images that offer you various memorable areas for placing your secret clicks	
Clicking on images	Try not to click within the same or adjacent areas	Oorchot et al., [6] showed that some users' secret were predictable. To reduce this, the user is advised to create their secret randomly. Specifically, they are not permitted to click on or within the same area (also applied to many images), advised not to create an easy to guess pattern (e.g. straight line) and encouraged not to click on obvious objects (e.g. edge, centre of each image).
	Try to click on various areas, not only on an obvious object	
	Please avoid predictable patterns (e.g. straight line, edges, central of images, etc)	

The next section of this paper highlights the methodology, followed by the results, discussion and conclusions.

2 Methodology

A graphical software prototype known as the Enhanced Graphical Authentication System (EGAS) was developed using Microsoft Visual Basic 2008. EGAS is an alternative graphical authentication employing a combination of both click and choice-based methods. In the EGAS software prototype, users are given the freedom to choose their preferred number of clicks (secret clicks), with the software assigning the number of images (secret images) they need to choose. Table 2 shows the combination of secret clicks and secret images.

Table 2. Click and Image details used in the software prototype

Secret click chosen	Secret image assigned	Image size/Tolerance
1	6	200x200 / 7x7
2	5	200x200 / 7x7
3	4	200x200 / 7x7
4	3	200x200 / 7x7
5	2	200x200 / 7x7

Two types of data collection were implemented; named as 'internal' and 'external'. Internal means the experimenter observed participants during trial (similar with the one to one usability testing) and they had to complete current task before proceeding to the next (controlled by the software prototype). Participants within the external group had to install the software prototype into their personal computer and use it for three weeks, with all of their activities recorded into a database (no means of control was enforced by the software prototype).

Participants for both groups (internal and external) had to register their details (username and secrets) in the software prototype, were then required to log into the software using their chosen secrets and finally provide feedback via a questionnaire. All tasks were done within the software prototype.

During the secret registration (enrolment), the GPG were first displayed to them (by which they had to acknowledge the GPG) before they chose their secret. During image selection, participants were able to choose images from 10 different themes (buildings, abstract, food, animals, flowers, view, people, sport, transport and fruits), with each of them consisting of 9 distinct images (arranged in 3x3 grids).

Participants within the internal group were asked to login three times, while the external group needed to login on four different days in week 1, two different days in week 2 and finally login once in week 3. This aimed to examine their familiarity and competency (e.g. login time, clicking accuracy, total attempts).

The trial was conducted over two months with the participants of the internal group recruited via an open call for volunteers within the authors' university. Participants of the external group were colleagues/friends of the author (external to the University) and invited via email, chat messengers and text messages.

The data were interpreted and reported into five main categories; namely number of attempt, timing, pattern, accuracy and finally users' feedback. The number of attempt looks upon participants' failure and success trials during both registration and login tasks while timing reports the time needed for these tasks. Pattern discusses the occurrences of 'hotspot' and 'hot-image', with accuracy mainly focuses upon the participants' ability to click on their secret clicks and finally users' feedback reports participants' perception on the questionnaire.

3 Results and Discussion

In total, there were 48 participants participated. Table 3 gives information for both groups highlighting the gender split and number of participants who had previously participated in graphical password studies [18].

Table 3. Participants' information

Demographic	Internal group	External group
Male participant	12	10
Female participant	18	8
Experienced using GA	10	2

3.1 Number of Attempt

3.1.1 Internal Group

Members of this group undertook 356 of authentication attempts. Of these, 94 logins were successful and 47 failed, 156 failed during registration and 66 were able to register successfully (note that software recorded two trials for each participant if they managed to register).

Participants who changed their click decided to choose the lowest click. Of the total seven participants who initially chosen three clicks on each image, five of them went to one click, while the remaining chosen two clicks. Moreover, all five participants who initially chosen two clicks and one participant who initially chosen four clicks also decided to choose one click.

During login, all participants within all click groups performed well where they managed to login, these results improved with experience. Only ten participants recorded a complete failure to login. There were six occurrences of failed attempts for login one, four occurrences for login two and only three occurrences for login three. The ability of participants to login with fewer failed attempts suggests participants performance improved with experience.

3.1.2 External Group

With eighteen participants within this group, the software recorded a total of 283 login attempts in week one, 61 trials for week two and finally 30 for week three. Of these, there were 92 successful logins for week one, 51 for week two and 20 for week three (note that there were participants who logged into the software more than was asked for).

Investigation of successful usernames who continued with the login tasks found mixed results. It was found only 12 participants followed the login interval task, with the remaining 6 participants using the software occasionally. For those who logged into the software according to specified tasks, 9 participants had chosen one click, 1 participant chose two clicks and 2 participants chose five clicks. Analysis has also found that 6 participants (who did not complete the login tasks) infrequently login during week one, with three of them logged twice for week two and finally all of them logged into the software in the third week. Five of them had chosen one click, while the remaining participant went for five clicks.

Only eight of the external group participants managed to register by using their first username. Of the remaining 10 participants who used a second username, six of them changed their secret click to the least click. Unless otherwise stated, most of the analysis for this group was based upon 18 participants who completed the specified tasks.

3.2 Timing

3.2.1 Internal Group

The time for participants to register and then log into the software prototype was recorded with the time during registration calculated from the point when they pressed the 'register account' button until to the result for registration is displayed. The time for login was calculated from when the participant started to enter their username until the last click for their secret images.

Table 4 shows participants' time (average, shortest, longest and standard deviation) during registration and three logins, in minutes, (m) and seconds, (s).

Table 4. Timing for the internal group

Click	Participant	Time	Registration	Login One	Login Two	Login Three
1	18	Average	5m 23s	24	20	18
		Shortest	1m 43s	15	11	9
		Longest	21m 58s	42	42	4
		SD	4m 43s	8	8	6
2	5	Average	10m 29s	40	35	27
		Shortest	2m 23s	28	25	18
		Longest	23m 33s	71	69	40
		SD	7m 56s	17	18	8
3	3	Average	9m 56s	39	33	33
		Shortest	5m 47s	36	23	22
		Longest	16m 46s	43	39	42
		SD	5m 58s	4	9	10
5	2	Average	2m 56s	26	20	23
		Shortest	1m 12s	24	19	21
		Longest	4m 40s	28	21	24
		SD	2m 27s	3	1	2

For all click groups, the registration time can be considered long due to the action of selecting images and then clicking on the chosen images. It can be reported that for all click groups, the time to login during login attempts one to three are significantly shorter. The study also found that participants do not immediately select their click area, often taking several seconds before they start clicking on it. This action is believed to be due to the small tolerances used, which suggests it could directly affect login time and security if the users were observed.

3.2.2 External Group

Table 5 shows the time for 12 participants as they managed to login according to the specified login intervals. L1 to L5 refers to the login times (measured in seconds, (s)) for week one, L6 and L7 are login times for week two and finally L8 refers to the login time for the third week. It was found that the login time across the three weeks varied, although with one click, participants showed little change.

Table 5. Timing for the external group

Click	Participant	Time	Register	L1	L2	L3	L4	L5	L6	L7	L8
1	9	Average	11m 17s	17	20	17	20	14	17	17	14
		Shortest	2m 15s	14	14	13	11	12	10	10	9
		Longest	47m 23s	23	39	28	37	22	31	43	21
		SD	14m 2s	4	8	7	10	3	6	11	3
2	1	Average	3m 22s	19	26	21	31	16	15	18	24
		Shortest	3m 22s	19	26	21	31	16	15	18	24
		Longest	3m 22s	19	26	21	31	16	15	18	24
		SD	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
5	2	Average	10m 2s	33	29	23	35	25	28	20	27
		Shortest	2m 39s	29	24	22	22	23	23	19	26
		Longest	17m 25s	37	34	23	48	27	32	20	28
		SD	10m 26s	6	7	1	18	3	6	1	1

3.3 Accuracy

As reported earlier, the numbers of failed attempts during registration were high. As a result, participants had to use other usernames and changed their preference secret click or image. The authors discovered two main errors associated with such scenario, as indicates below.

a) Participant was unable to click within the allowable tolerance.

b) Participant did not click in sequence order, as the result of forgetting their secret order or areas.

From the data for both internal and external groups, it can be revealed that errors during both registration and login were correlated with participants who selected more clicks. In specific, there were slightly more participants who made tolerance errors than order errors. This is probably due to the software prototype using a small click tolerance.

Particularly within the external group, participants were unable to click accurately when they first started using the prototype. However, they managed to click within the clickable areas as they became familiar with using the software and understood what they needed to accomplish.

3.4 Pattern

Patterns are created during image selection when participants chose the same images (in the case of changing username of click), gender skew selection (e.g. men choosing sports car while women chose flowers) and following image order (e.g. participants choosing the first image in each theme). Moreover, patterns during the click selection are created when participants clicked on the same area across all images, producing obvious shapes or clicking their secrets in a straight line (e.g. top, bottom and left side of image area), and clicked on the image that appeared to be offering a pattern. Results for both groups are reported together within this section as they used similar software prototype.

With the internal group, the study found the majority of participants who changed their username or secrets (click or image) used their previously chosen images. One participant from the five clicks group used both of his previous images while two participants from the two clicks group used three and one of their previous images respectively. Of all the participants from the one click group who changed their username or clicks, only one did not used their previous image. Specifically for the one click group, two participants used four of their previous images while the others were ranging from one to three. In addition, it was also found one of these participants selected the first image for each theme as their secret images.

The external group also used their previous secret images with one participant using all of their previous images, with six other participants using between one to two of their previous chosen images .It was also found that two participants of the one click group chose their images in sequence (choose the first six themes); however their chosen images were different with each other.

Table 6. Image popular with their associated number of male and female

Theme	Number of participants choosing popular image	Male	Female
Buildings	11	3	8
Abstract	12	6	6
Food	8	4	4
Animals	7	3	4
Flower	10	4	6
View	14	7	7
People	5	2	3
Sport	13	8	5
Transport	4	1	3
Fruits	7	2	5

Table 6 presents the number of participants who chose popular images for each theme. It was found that the view and sport themes are the most popular, with the transport and people themes as the least popular selection.

Although it was found that a number of participants clicked within similar areas when creating their secret clicks, such action was eliminated due to the software prototype preventing participants from clicking on the same area within multiple images. Analysis was carried out to examine the area of clicking for participants who chose more clicks and although it can be reported that participants with two or three click groups create less obvious pattern, participants of the five clicks group clearly create patterns. The authors deduced that such scenarios are related to the images themselves, which clearly offer a pattern to be created.

Analysis was also done to examine the click areas in popular images for each theme. Analysis on the one-click group who chose the most popular image revealed that ten of the twelve participants who chose popular images in the sports theme clicked on the three most popular areas (see left side of the fig. 1), with seven out of twelve participants who chose the most popular image for the 'view' theme clicked on the same area (see right side of the fig. 1). Equally, all other popular images have shown a pattern where participants clicked on similar spots.

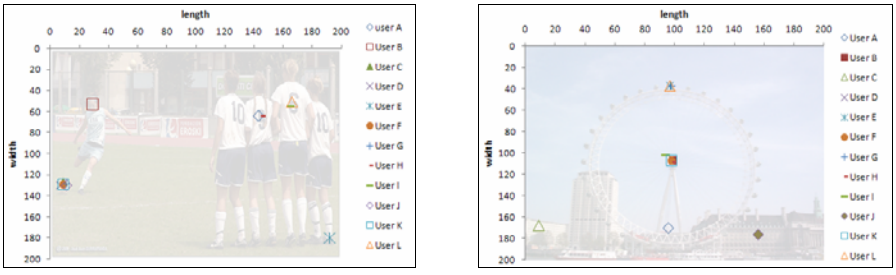


Fig. 1. Participants click areas for the popular image of the 'sport' (left) and 'view' (right) themes

From the collected data, the authors summarised that participants who chose more clicks tended to create patterns during their clicking task, while the existence of pattern during image selection was unidentified. Meanwhile, participants who chose more images (fewer clicks) tended to create patterns during both image and click selection. Patterns where users chose the first or last image for each theme was also reported. Although the authors' approach of not implementing restrictions for image selections and depending solely upon the guidelines is less effective then the introduction of guidelines. The GPG itself has resulted in the reduction of gender bias image selection and image order patterns.

It can, however, also be said that the restrictions together with the guideline during secret click selection played a minor role during the click selection task. Although not representative, participants with a higher number of clicks created more patterns (possibly as it is easier for them to remember), with analysis towards one click participants revealing the existence of hotspot.

3.5 Users' Feedback

A Likert five points scale rating was used to obtain participant feedback with the lowest score indicating participants' agreement with the statements while the highest score indicating disagreement. Table 7 reports the mean score of feedbacks for the first three questions within the internal group.

Table 7. Questionnaire results

Question	Mean score
Perception towards graphical password guidelines (GPG)	1.6
Perception towards restrictions	1.8
Perception towards combining GPG with the restrictions	1.8

When asked about participants average login time (with the software prototype displaying their average login time), it can be revealed that twenty participants found their login time were acceptable, with eight unacceptable. Seventeen participants agreed that their total registration time was acceptable while the remaining eleven disagreed. With the statement on the optimum combination of image and click, twenty two of the participants felt that having more images was more memorable than

having more clicks, while five participants felt that the balance between both click and images were still memorable.

Participants who were new to the graphical method felt the method could be very useful and provided excellent protection. However, the majority of the participants who were involved in the previous trial felt that having larger clickable areas was more usable. In addition, they felt that having more clicks could be troublesome as they had to memorise too many spots and finally all participants agreed that in order for them to perform better, they needed to become more familiar with the method.

4 Conclusions and Future Work

This paper presented an investigation of the practicability of giving guidelines to a user before they chose their secrets for a graphical authentication system as well as evaluating user attitudes and opinions to the enhanced techniques.

During the registration task, participants struggled to click accurately within the allowable click tolerance and those who chose more clicks often failed to click in the correct order. As the result, they had to change to create new accounts or change to fewer clicks. The login task had shown improvement as they managed to login with fewer failed attempts, and the time to login to the software prototype was reduced marginally across login interval. The above findings reflect participants' familiarity with the software prototype as they used the software regularly.

Introducing guidelines to the participants before they start selecting their secrets had obtained positive perception from the majority of participants. However, this study has shown that guidelines on their own cannot guarantee the security and safety of the method itself. This is because participants used their previous images and created secret clicks using easy to remember spots, which resulted in predictable click-areas. By combining the introduction of guidelines with restrictions, user behaviour can be controlled to safeguard the method. This was proven where cases such as clicking on similar areas within the same or multiple images and where creating predictable pattern were reduced.

Finally, this paper has shown that the click patterns created by users who chose more clicks had a direct relationship with the nature of the image itself. It could be said that the introduction of guidelines gave no effect on participants' usability performance, but might give positive or negative effects on the security. The study also suggests that using one click per image is an ideal combination. This is because using one click per image requires less memorisation (i.e. it is more suitable for users with multiple accounts), less time to authenticate, convenience and significantly safer from predictability. The study also suggests that using a small tolerance without giving sufficient opportunity for familiarity to the user could result in a lack of usability of the proposed method.

It is suggested that future work could include a larger and more varied participant based for conducting significance testing to validate the collected data, testing different restrictions with the GPG, further evaluation of the claim that 'one click per image is better' and evaluating the technique known as the 'graphical-passwords-strength-meter', for safer secret creation based upon feedback from the system itself.

References

1. De Angeli, A., Coventry, L., Johnson, G., Renaud, K.: Is a picture really worth a thousand words? Reflecting on the usability of graphical authentication systems. International Journal of Human Computer Studies 63(2), 128–152 (2005)
2. Chiasson, S., Oorschot, P.C.V., Biddle, R.: Graphical password authentication using cued click points. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 359–374. Springer, Heidelberg (2007)
3. Hinds, C., Ekwueme, C.: Increasing security and usability of computer systems with graphical password. In: ACM Southeast Regional Conference, Winston-Salem, North Carolina, USA, pp. 529–530. ACM, New York (2007)
4. Chiasson, S., Forget, A., Biddle, R., Oorschot, P.C.V.: User interface design affects security: Patterns in click-based graphical passwords. International Journal of Information Security 8(6), 387–398 (2009)
5. Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., Memon, N.: PassPoints: design and longitudinal evaluation of a graphical password system. International Journal of Human Computer Studies 63, 102–127 (2005)
6. Oorschot, P.C.V., Salehi-Abari, A., Thorpe, J.: Purely automated attacks on Passpoints-style graphical passwords. Transactions on Information Forensics and Security 5(3), 393–405 (2010)
7. Davis, D., Monrose, F., Reiter, M.K.: On user choice in graphical password schemes. In: Proceedings of the 13th USENIX Security Symposium, California, USA, August 9-13, pp. 1–11. USENIX Association (2004)
8. Tullis, T.S., Tedesco, D.P.: Using personal photos as pictorial passwords. In: CHI 2005 Extended Abstracts on Human Factors in Computing Systems, Portland, Oregon, USA, pp. 1841–1844. ACM, New York (2005)
9. Everitt, K.M., Bragin, T., Fogarty, J., Kohno, T.: A comprehensive study of frequency, interference, and training of multiple graphical passwords. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, Boston, MA, USA, pp. 889–898. ACM, New York (2009)
10. Dirik, A.E., Memon, N., Birget, J.-C.: Modelling user choice in the Passpoints graphical password scheme. Paper presented at the Symposium on Usable Privacy and Security, Pittsburgh, PA, USA, July 18-20 (2007)
11. Gołofit, K.: Click passwords under investigation. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 343–358. Springer, Heidelberg (2007)
12. Golofit, K.: Picture passwords superiority and picture passwords dictionary attacks. Journal of Information Assurance and Security 2, 179–183 (2007)
13. Peach, S., Voster, J., Heerden, R.V.: Heuristic Attacks against graphical password generators. In: Clarke, N., Furnell, S., Solms, R.V. (eds.) Proceedings of the South African Information Security Multi-Conference (SAISMC 2010), Port Elizabeth, South Africa, pp. 272–284. University of Plymouth (2010)
14. Lin, P.L., Weng, L.T., Huang, P.W.: Graphical password using images with random tracks of geometric shapes. In: Proceedings of the 2008 Congress on Image and Signal Processing, pp. 27–31. IEEE Computer Society, Los Alamitos (2008)
15. Harada, A., Isarida, T., Mizuno, T., Nishigaki, M.: A User Authentication System Using Schema of Visual Memory. In: Ijspeert, A.J., Masuzawa, T., Kusumoto, S. (eds.) BioADIT 2006. LNCS, vol. 3853, pp. 338–345. Springer, Heidelberg (2006)

16. Hayashi, E., Dhamija, R., Christin, N., Perrig, A.: Use Your Illusion: secure authentication usable anywhere. In: Proceedings of the 4th Symposium on Usable Privacy and Security, Pittsburgh, Pennsylvania, pp. 35–45. ACM, New York (2008)
17. Chiasson, S., Forget, A., Biddle, R., Oorschot, P.C.V.: Influencing users towards better passwords: persuasive cued click-points. In: Proceedings of the 22nd British HCI Group Annual Conference on HCI 2008: People and Computers XXII: Culture, Creativity, Interaction, Liverpool, United Kingdom, vol. 1, pp. 121–130. British Computer Society (2008)
18. Jali, M.Z., Furnell, S.M., Dowland, P.S.: Assessing image-based authentication techniques in a web-based environment. Information Management & Computer Security 18(1), 43–53 (2010)
19. Chiasson, S., Biddle, R., Oorschot, P.C.V.: A second look at the usability of click-based graphical passwords. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, Pittsburgh, Pennsylvania, pp. 1–12. ACM, New York (2007)

A Case Study in Practical Security of Cable Networks

Amir Alsbih[1], Felix C. Freiling[2], and Christian Schindelhauer[1]

[1] Albert-Ludwigs-Universität Freiburg, Germany
[2] Universität Mannheim, Germany

Abstract. Cable networks are complex systems that have evolved over years and in which new features like Internet access and Voice over IP (VoIP) have been integrated. We argue that threat models must evolve together with such systems and show that inadequate threat models can be used to explain known and unknown vulnerabilities in today's cable networks. We do this by demonstating an attack on the DOCSIS provisioning standard in cable networks. By exploiting this weakness, an attacker can hijack VoIP accounts. We also show how to mitigate the attack.

1 Introduction

1.1 Motivation

Cable networks were initially deployed as a low-cost means to broadcast television programs to customers, first analog and then also digital. Today, almost all cable networks have been re-engineered so that they are able to additionally transport arbitrary digital data both to and from the end user. The number of Internet users in Germany and elsewhere that access the network in this way is rising sharply [5]. Within the system of cable networks there are three main stakeholders: First of all, there is the user (customer) who wishes to enjoy digital media and network access in high quality at low cost. Second, there is the cable network provider (CNP), a company running the physical cable network infrastructure. Third, there is the Internet service provider (ISP) who provides access to the global Internet. Traditionally, the context and mindset of the CNP is a closed, physical network with static functionality. Since the services of an ISP were added to the portfolio of the CNP only rather recently, the original mindset of the CNP may be reflected in the way the ISP is managed. Since the Internet is an open, virtualizable and dynamic network, there is a potential for misconceptions regarding security assumptions that can lead to many surprises. In this paper we show that this is in fact the case.

1.2 Context

Technically, the user's *cable modem* acts as bridge between the customers home network and the backbone of the ISP. The cable modem is a rather simple device

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 92–103, 2011.

that downloads its configuration after every reboot from a server at the ISP. This is called *provisioning* and the relevant standard is the *Data Over Cable Service Interface Specification* (DOCSIS) [7]. So there is a way to access the ISP servers from the customer's home networks and therefore the ISP has to secure the provisioning process to restrict the possibilities of a service abuse. In this paper, we document a weakness in the way the provisioning process is handled today.

1.3 Related Work

There is relatively little work that investigated cable networks security from an academic viewpoint. Existing work mainly comes from industrial or rather applied forums and deals with possibilities of service theft, e.g., possibilities of achieving higher service bandwidth without paying for it (so-called "uncapping") by manipulating the cable modem [1–3, 11, 15]. Other threats to ISPs have been formulated as well [1] and refer to attacks on confidentiality and weak endpoints: The classical attack to confidentiality of network traffic through eavesdropping is omnipresent in shared medium networks like the cable network is. Since the downstream traffic is broadcast across the shared medium, network providers have to enforce rigid access control techniques at the endpoints of the network to ensure confidentiality. A related threat is network access through stolen authentication credentials, e.g., cloning of MAC addresses, a problem that is hard to tackle in networks where endpoints are under complete control of the customer [2]. In this paper, we give another example for this fact. Since one of the main applications of cable networks is digital telephony (voice over IP, VoIP), attacks on the corresponding protocols like the Session Initiation Protocol (SIP) have also been investigated in the cable network environment [3, 8]. These attacks include tampering of SIP message bodies like malformed SIP messages, hijacking dedicated SIP accounts or interrupting sessions by injecting fake messages into the network traffic. While being relevant to VoIP technology, they are only specific to cable networks as far as these attacks use access techniques that only exist in cable networks. In this paper, we present such an attack on SIP that is specific to cable networks.

1.4 Contributions

In this paper, we describe the context and specifics of the system of cable networks as an evolving complex system that is in regular use all over the world. We show that different mindsets and threat models can be used to explain past and present vulnerabilities in such networks. As a confirmation, we present a new attack on VoIP in cable networks that allows an attacker to extract SIP credentials and therefore misuse the VoIP system in such networks. More specifically, our attack exploits the DOCSIS provisioning requirement that every cable modem needs a provisioning file that contains the configuration of SIP credentials. By gaining access to the management network and partial exhaustive search of the namespace of configuration file names, we are able to extract SIP credentials and take over telephone acounts of other customers. We also show how this attack

can be mitigated by adjusting network management policies in cable networks. At the time of writing, our attack was possible in one major (cooperating) cable network in Germany. Since many other CNPs all over the world use the same hardware and software configurations, we believe that the attack is relevant not only in Germany. However, the wish to point out that the attack is just a vehicle to transport the another insight, namely that threat models must be checked regularly. And if they turn out to be unrealistic, they (and all affected security procedures) must be adapted.

1.5 Paper Outline

This paper is structured as follows: We give a brief introduction into cable network technology in Sect. 2. In Sect. 3 we show how threat models for cable networks have evolved in the past and the effect of this process on the provisioning process. We present the resulting attack in Sect. 4 and possible countermeasures in Sect. 5. We conclude in Sect. 6.

2 Background

In this section we give a brief introduction into the basics of modern cable networks.

2.1 System Overview

The cable network is a complex system consisting of multiple interconnected networks (see Fig. 1). The customer network connecting the end user devices

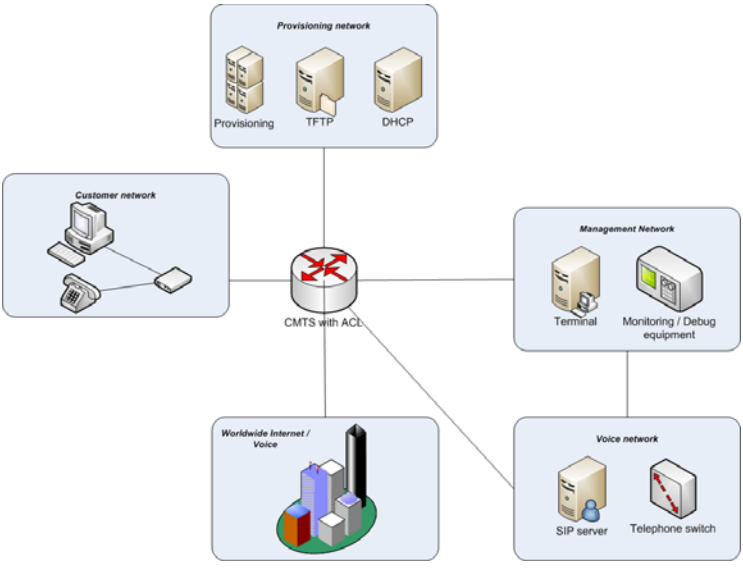


Fig. 1. Cable network reference figure

with each other is a broadcast network that can be used to transport analog and digital signals in an integrated fashion over distinguishable frequency bands. It can be thought of as a long wire to which many receiving stations can be connected, similar to the early Ethernet technology (10BaseT). Technically, the customer network is a hybrid network consisting of optical fiber and coaxial cables that is connected in a tree-like topology. Consequently, this network is called *hybrid fiber coax* (HFC).

2.2 Physical Aspects and Frequency Bands

Inside the HFC network, the frequency spectrum is divided into channels. The channels can be divided into *downstream* (towards end user) and *upstream* (from end user). Originally, all channels were downstream and carried either radio or television signals. Subsequently, frequencies were defined for data (i.e., Internet) communication. Since this communication is bidirectional, the CNP will need to provide at least one channel in each direction. On top of this digital channel, digital telephony services (VoIP) can be offered. For Internet communication, the digital signal has to be modulated on to and demodulated from the physical medium at each connected station. This is done at the side of the end user by a *cable modem*. At the side of the cable network provider, this is done by the *cable modem termination system* (CMTS). This is similar to how DSL technologies work over the telephone network.

2.3 The Interfaces of the Cable Modem

Internally, the cable modem consists at least of two interfaces, each with its own MAC address:

- The first interface is used for remote management of the cable modems from the side of the CNP. It has a unique MAC address called *C-MAC*.
- The second interface is used for realizing the Internet service for the customer. It's MAC address is called *E-MAC*.

After the cable modem has been connected to the cable and turned on, the cable modem will configure itself. This process is called *provisioning* and will be explained later, since it is in the center of our attack. The result of a correct provisioning process is that both interfaces will receive its own IP address, each IP address being in a seperate IP address range.

2.4 CMTS and Access Control

If a cable modem is provisioned and tries to communicate, every communication request of the cable modem is handled by the CMTS. The CMTS plays the role of a central router, guiding network packets from the customer network to the Internet and vice versa. At the same time, the CMTS acts as a firewall, enforcing filter rules to, for example, separating the network traffic from the different IP address ranges belonging to the different interfaces of the cable modem. Filtering

is even performed on packets that are "routed back" into the customer network. This happens, for example, if two cable customers communicate with each other. So even if the cable network environment is a shared medium every upstream communication is only possible over the CMTS. The filtering rules of the CMTS are one of the most important parts of cable network security. For example, one of the most important rules is one that forbids normal users (via their E-MAC) access via SNMP to the management interface of the cable modem (C-MAC) of other customers. Without this rule it would be possible for every customer to access (and manage) the cable modem of other customers via SNMP. Since the filtering rules on the CMTS are the only reliable way to restrict the customer's communication abilities, it is important to invest a lot of time into the right setup, making sure that the customer is only able to act in the way intended by the CNP [9, 14].

2.5 IP Layer

As mentioned above, the cable modem and the CMTS function as endpoints to transport data over the physical HFC network. The CMTS routes the IP data from the fast Ethernet backbones of the ISP to the cable network and vice versa. The cable modem works as a bridge between the network of the ISP and the local area network of the customer (see Fig. 2).

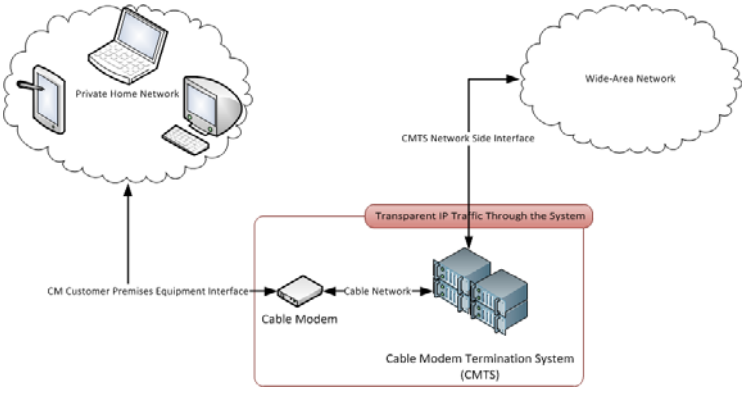


Fig. 2. IP traffic via cable modem and CMTS based on[6]

2.6 VoIP via SIP

The *Session Initiation Protocol* (SIP) is the most commonly used VoIP protocol today. SIP phone calls use two different protocols, SIP for the connection handling of the calls, and *Realtime Transport Protocol* (RTP) for the voice stream. SIP was designed for the IP world, and since IP addresses are not as static as telephone numbers, the clients have to register themselves at a SIP server, to let the SIP server know where it should route the incoming call to. To register an account on a SIP server, the customer needs (1) his own phone number, as well

as (2) the corresponding username and (3) the password for that phone number. He also needs to know the SIP server managing the connection. Note that these three items have to be known to the cable modem in order to allow seamless telephone service over VoIP for the cable network customer. Therefore, these credentials are contained within the configuration file during the provisioning process (see below).

3 Different Threat Models and Their Effects on the Provisioning Process

A *threat model* is a precise description of the possible threats to the system [16]. It usually consists of a set of security issues a system designer cares about together with a set of expected attacks. Often, threat models only exist implicitly in the mindset of the people working at the network operator and are therefore not documented within organizations. In such cases the threat model used in an organization can only be inferred through interviews and from analyzing existing security mechanisms.

3.1 Traditional Threat Model of the CNP

Traditionally, the context and mindset of the CNP is a closed, physical network with rather static functionality. Before upstream data communication was possible, the cable network was a pure "broadcast" network. The endpoints (antenna sockets in houses) were usually protected by physical means like tamper-evident seals. This was also the way how access control to the cable network worked. Since the transmitted data was the same for everyone and the selection of which channel to watch was performed at the endpoint (the television set), there were also no real privacy or confidentiality problems. Possible attacks involved only physically breaking the seal of the endpoint and accessing the service without paying.

3.2 Adapted Threat Model of the CNP/ISP

The threat situation changes dramatically if individual communication is handled via the cable network both upstream and downstream. The typical threat model used by CNP/ISP in these scenarios, however, is very similar to the original threat model. As mentioned above and from our experiences, the threat model is usually not explicitly documented. So we inferred the following assumptions from interviews and an analysis of the literature on known attacks [1–3, 11, 15]:

- The endpoint is physically protected. This means that only original cable modems are attached to the cable endpoints and these cable modems always correctly follow the provisioning process.
- The end users are untrustworthy, i.e., they may send and receive arbitrary packets via their E-MAC to/from the Internet. This implies that the management network (accessed using the C-MAC) needs good protection from the user network (accessed using the E-MAC).

The second point is realistic and the main reason for the complex filtering rules within the CMTS. The first point, however, does not hold in todays networks and can be exploited in most cable networks today, as we now explain.

3.3 The Provisioning Process and Its Weaknesses

The DOCSIS standard describes the steps each modem has to fulfill to register itself on the cable network. If a step in the process fails, the modem has to repeat the step until it succeeds. Since it is up to the CNP where he will place the digital channels that are in use for realizing the Internet service, the first step of the cable modem is a large frequency scan to search for the downstream channel. After that, the cable modem will get the parameters for the upstream by searching for a special packet in the downstream channel called *upstream channel descriptor*. Since the cable modem now has knowledge about both the down- and the upstream channels, the modem now has to synchronize itself to the channels in a step called ranging. In this step, the cable modem adjusts the timing, power, and frequency to balance the network delay. Subsequently, the cable modem establishes IP connectivity. Therefore it sends a Dynamic Host Configuration Protocol (DHCP) discover packet with option code 60 (vendor class identifier) for the C-MAC and a normal DHCP (without option 60) for the E-MAC interface. The cable modem listens for a DHCP offer packet that contains the needed data. Option 60 of DHCP allows the interfaces to tell the DHCP server which kind of network devices they are [4] by attaching a message to the DHCP request. This is used to ensure that every interface gets an IP address in a separate IP address pool. The DHCP offer for the cable modem contains an IP address that is assigned to the cable modem management part (C-MAC). Usually, this IP address comes from the cable modem address pool 10.61.0.0/16 [7]. The DHCP offer also includes the IP address of a TFTP server in the management network and the name of a configuration file residing on the TFTP server. The content of a typical DHCP offer is shown in Fig. 3. The client IP address (assigned to the C-MAC) is the entry in the field "your client IP address", in this case it is 10.61.151.101. The name of the TFTP server that hosts the configuration file for this cable modem is contained in the field "Next server IP address" and is 172.30.*.* in this case. The name of the configuration file itself is encoded in the "Boot file name" field.

As next step, the cable modem will download the configuration file from the TFTP server. After this step the modem has to send the file to the CMTS in a step that is called *transferring the operational parameters*. This has to be done to authenticate the modem as modem of the CNP. If the modem is in the database of the CNP, the CMTS sends a message to the modem that it has passed registration. Now the modem is fully provisioned and able to act as bridge between the cable network and the LAN of the customer. A sample configuration file is shown in Fig. 4 where critical data (like passwords) has been sanitized. The SIP username ("0305338890") and the SIP password ("ABCDE123456") are directly stored in cleartext. This shows that access to configuration files opens complete access to a SIP account.

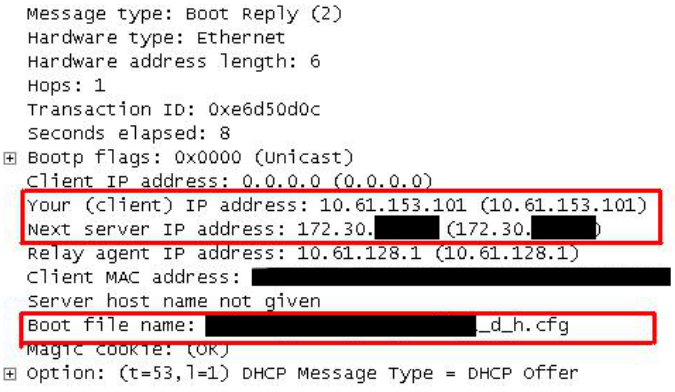


Fig. 3. Excerpt from the DHCP offer sent by the DHCP server within the configuration process (captured and visualized using Wireshark, identifable data is obfuscated)

```
PCMA
comm1
public
comm1
@mtaprov
comm1
comm1
comm2
5g21wm7sdl
comm2
@mtaprov
comm2
comm2
'x[0-9]*.[t#]|11[025]|[*#][2-9]x|[*#]1xx
&My Small Company
My-Small-Company.com
SIP.Registrar.IP
SIP.Registrar.IP
ABCDE123456
0305338890
0305338890
0305338890
```

Fig. 4. Example DOCSIS configuration file extracted with strings

4 Attacking the Provisioning Process

There are many reasons for an attacker to steal configuration files, but only the SIP credentials are profitable for an attacker. Therefore, we aim to steal the configuration files that contain SIP credentials from the provisioning servers. With the SIP credentials, the attacker can make free telephone calls, hijack and spoof phone calls, and do anything that the real owner of the SIP account can do.

4.1 The Attack

The steps of an attacker are as follows:

1. In a first step the attacker fakes the MAC address of his own computer using standard tools [13] for anonymity and maybe also to avoid access control restrictions in the CMTS (for example, if only certain vendors are allowed).

Then he attaches his own computer to the HFC network to the LAN port of the cable modem.

2. Now the attacker accesses the provisioning network. The attacker has to spoof the device information of his computer in a way that the DHCP server that is responsible for the cable modems "thinks" that the attacker is in fact a part of the cable modem. This can be realized by configuring the DHCP client to send the right form of option 60 with his DHCP request. One correct form of this option is for example to set the vendor class identifier to the value `"pktc1.0"` [7]. This tells the provisioning server that the device sending the request is a cable modem that is only capable of the DOCIS 1.0 provisioning process. After issuing this request using appropriate tools [4], the attacker will receive the corresponding DHCP response including an IP address inside the HFC access network. The DHCP response also includes the IP address of the TFTP server hosting the configuration files, and a route through the CMTS to the provisioning network.
3. Depending on the MAC address of the attacker, the name of the configuration file within the DHCP response will be some default configuration file name pointing to a location on the TFTP server that will not contain anything important. If by chance, the MAC address is known to the TFTP server, it will point to a configuration file containing the credentials of the corresponding user. Since the name scheme of the configuration files is up to the CNP, it could require some reverse engineering to find out the mapping of MAC addresses to configuration file names. One approach, for example, is to sniff the provisioning process by using the real MAC address of the attacker. In practice we observed different naming schemes. One scheme concatenated the MAC address with the suffix `d_u.cfg`. Using this knowledge about the configuration file naming scheme and with access to the TFTP server, the attacker can easily enumerate configuration file names by using tools like *TFTP brute* to brute force MAC.
4. After the successful download of one or more configuration files, the attacker can extract the SIP credentials from the configuration files easily, e.g., by using the Linux command `strings`.

Now it is possible to abuse that SIP account.

4.2 Why Is the Attack Possible?

There are two specific points in the attack that are critical for its success. First, the names of configuration files are deterministic and can be enumerated. This problem can be easily fixed by giving configuration files a new random name when they are distributed. In a sense, this also ties a specific configuration file to a specific MAC address. But even if this is done, the second problem remains, namely that end users can impersonate any other device by spoofing their MAC address. In summary, the attack exploits exactly those weaknesses that result from an inadequate threat model. The main point is the inadequate assumption that endpoints are physically protected and therefore impersonation is impossible. That this attack exists is surprising since it is well known how easy it is

to attach computers instead of cable modems to the network endpoints. Similarly, it is usually possible to reprogram cable modems by installing manipulated firmware. As shown in the attack, this opens the path for MAC address spoofing and impersonation of specifically weak end devices.

5 Countermeasures

There are many obvious technical countermeasures to the attack described above. The basis for good countermeasures, however, is a more realistic threat model.

5.1 Adapted Threat Model

The adequate threat model for cable networks takes into consideration that the CMTS is the last (physically) controllable point in the cable network and that the cable modem is an untrustworthy device that could do anything and act different from the way that is expected. In particular, it is not possible to bind service usage to particular physical endpoints as it can be done in classical telephone networks.

5.2 Technical Countermeasures

Binding service usage to particular users is known as the problem of authorization. A prerequisite for authorization is authentication. The DOCSIS standard specifies a set of features that can enforce authentication within the provisioning process (BPI+ and DOCSIS shared secret [1, 10]). Most of these features are, however, not used or only optional and not enforced in practice. These features at least prevent other known attacks such as uncapping of cable modems, clear text network traffic on the downstream side and normal protocol attacks that are not specific to cable networks by not only enforcing authentication but also encryption. The fact that they are almost never turned on points again to an insufficient threat model but possibly also can be explained by the cumbersome effort to set up a public key infrastructure and equip modems with certificates. While enforcing authentication, both BPI+ and DOCSIS Shared Secret do not prevent the mass downloading of configuration files. One approach to prevent this is to hide the location of the TFTP server from the customer. By using *cable dynamic-secret mode* (DMIC), the CMTS will change the DHCP offer in the way that it will point to the CMTS and not to the TFTP server. The CMTS then will download the configuration file from the real TFTP server and insert a HMAC based on a onetime password and a cryptographic hash of the file. The modem then only is allowed to register itself if it contains the correct HMAC. The important aspect is that the TFTP server is hidden from the customer. This makes it impossible for the attacker to brute-force configuration files and extract the SIP credentials [12].

A similar result can be achieved by using randomization in the following way: Whenever a cable modem sends a DHCP request, the name of the configuration

file is chosen in a random fashion and uploaded to the TFTP server. The space from which the filename is chosen must be large enough so that it is hard to guess real filenames (e.g., a 64 bit number). But this method has not been standardized in DOCSIS yet. The DOCSIS standard also does not specify any procedures for the secure provisioning of cable modems that use VoIP since there is no well-tested and accepted procedure for this yet. It is not clear whether there will be a clean and working solution in the near future. Therefore, at least a fast detection approach for such attacks has to be implemented.

6 Conclusion

As shown in this paper and in the literature, the security problems of cable networks are not only the result of a weak DOCSIS standard, it is the way how DOCSIS is "implemented" within an organization that causes the attack vectors to persist. Many security features such as BPI+ are often not enabled. Since such actions are consistent with the current threat model that assumes the user endpoint as trustworthy, we have argued that the risks in modern cable networks are mainly due to inadequate threat models at the side of the CNP. While this point is true in general for all systems that allow attacks, cable networks offer a particularly interesting case study because they show how adding certain features (Internet access) to a secure system (TV cable networks) results in an insecure system if the threat model does not evolve too.

Acknowledgments. We thank Andreas Dewald and Martin Mink for helpful comments on a previous version of this paper.

References

1. Security on Data-over-Cable Systems: DOCSIS, BPI+ and Beyondm (November 2000), http://www.3com.com/other/pdfs/infra/corpinfo/en_US/50301102.pdf
2. Hacking the Cable Modem: What Cable Companies Don't Want You to Know. No Starch Press, San Francisco (2006)
3. PacketCable 2.0: Security Technical Report. Technical Report PKT-TR-SEC-V05-080425, Cable Television Laboratories, Inc. (April 2008)
4. Alexander, S., Droms, R.: DHCP Options and BOOTP Vendor Extensions. RFC 2132 (1997)
5. Bundesnetzagentur. Tätigkeitsbericht 2008/2009 Telekommunikation (December 2009)
6. Cable Television Laboratories, Inc., Cable Modem to Customer Premise Equipment Interface. Technical Report CM-SP-CMCI-C01-081104 (November 2008)
7. Cable Television Laboratories Research Consortium. DOCSIS Website (2010), http://www.cablelabs.com/cablemodem/
8. Endler, D., Collier, M.: Hacking Exposed VoIP: Voice Over IP Security Secrets & Solutions, 1st edn. McGraw-Hill, Inc., New York (2007)
9. Johns, M.S.: DOCSIS Cable Device MIB Cable Device Management Information Base for DOCSIS compliant Cable Modems and Cable Modem Termination Systems. RFC 2669 (1999)

10. Latini, P.S.: Avoiding Piracy in DOCSIS Networks. Canitec Conference and Exhibition (April 2010)
11. McKelvey, J.: Combating security risks on the cable IP network. Cisco Systems, Inc., Whitepaper (June 2002)
12. Millet, M.: Theft of Service — Inevitable? CableFAX: The Magazine (December 2005)
13. Pahwa, P., Tiwari, G., Chhabra, R.: Spoofing Media Access Control (MAC) and its Counter Measures. International Journal of Advanced Engineering & Application (January 2010)
14. Raftus, D., Cardona, E.: Radio Frequency (RF) Interface Management Information Base for Data over Cable Service Interface Specifications (DOCSIS) 2.0 Compliant RF Interfaces. RFC 4546 (2006)
15. Shah, N., Kouvatsos, D., Martin, J., Moser, S.: A Tutorial on DOCSIS: Protocol and Performance Models. In: International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks (July 2005)
16. Swiderski, F., Snyder, W.: Threat modeling. Microsoft Press, Redmond (2004)

Ceremony Analysis: Strengths and Weaknesses

Kenneth Radke, Colin Boyd, Juan Gonzalez Nieto, and Margot Brereton

Information Security Institute and School of Design,
Queensland University of Technology, Australia
{k.radke,c.boyd,j.gonzaleznieto,m.brereton}@qut.edu.au

Abstract. We investigate known security flaws in the context of security ceremonies to gain an understanding of the ceremony analysis process. The term security *ceremonies* is used to describe a system of protocols and humans which interact for a specific purpose. Security ceremonies and ceremony analysis is an area of research in its infancy, and we explore the basic principles involved to better understand the issues involved. We analyse three ceremonies, HTTPS, EMV and Opera Mini, and use the information gained from the experience to establish a list of typical flaws in ceremonies. Finally, we use that list to analyse a protocol proven secure for human use. This leads to a realisation of the strengths and weaknesses of ceremony analysis.

Keywords: Ceremony, EMV, HTTPS, Opera Mini, security, privacy, provable security, humans.

1 Introduction

In 1993 Bellare and Rogaway introduced a model for reductionist security proofs for cryptographic key exchange protocols [4]. Since this time, many cryptographic protocols have been accompanied by a reductionist security proof.

A reductionist security proof means that the security of the protocol is reduced to a known *hard* mathematical problem, such that if an advantage is achieved over the protocol, then there will be some significant advantage over the known hard problem. If a protocol is proven secure in this way then, as long as the "hard" mathematical problems remain sufficiently hard, the protocol is unbreakable within the defined security model.

Unfortunately, many protocols so proven to be secure in theory, have been found to be insecure in practice, when deployed in the real world. This inequality between the theoretical security and the actual security can be traced back to a deficiency in the security proof model. The mathematical security models while useful, especially for examining the security of a protocol in isolation, do not take into account the wide range of side channel attacks, social engineering, and interfaces to other protocols and the environment, which occur in the real world.

In 2007, Ellison wrote that a more robust method for examining the security of a protocol was to consider a security *ceremony* [7]. Ellison wrote with reference to network protocols, but we can extend that work to any group of protocols.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 104–115, 2011.

A security *ceremony* may be described as *protocols in their context of use*. For example, the protocol HTTPS provides a connection secure from eavesdroppers between two nodes on a network. However a security *ceremony* would include a user, viewing a website on their computer, and using HTTPS via their web browser running on the computer to securely connect to another computer on the network. We will show that while ceremony analysis is powerful enough to capture known attacks, each use case of a given set of protocols is a new ceremony and requires its own ceremony analysis.

1.1 Related Work

The concept of a ceremony was developed earlier than 2007 [8]. In the years since 2007, there has been an increasing trend to meld information security with the social sciences, as indicated by conferences both in the U.S.A[1] and in Europe[2]. This multi-disciplinary approach brings into context the human usage of information security systems. As Shostack and Stewart state, "... our approach to information security is flawed" and "the way forward cannot be found solely in mathematics or technology" [18].

Although little progress has been made regarding ceremonies since 2007, a number of researchers in different areas have agreed that ceremony analysis is a promising research direction. These research areas include formal methods, network security, and applied cryptography.

In the formal methods' security community, there has been a call to include parts of ceremony analysis in the formal methods' analysis of protocols [12]. This work has been further developed in Martina et al's more recent work in the PKI context [13]. Martina et al used the verification method outlined by Ruksenas et al. [16,17], adapted using Bella's goal availability principles [2], to address the open question that Ellison posed as to how to model human behaviour.

In the network security community, the concept of a ceremony has been used to describe protocols which include humans, and thus to create more robust security ceremonies [11]. Karlof et al. describe a concept of *conditioned-safe ceremonies*, based on a *defence-in-depth* approach adapted from the human reliability community. Central to their approach is the use of *forcing functions* whose property is to prevent a user from proceeding, until a critical step is completed.

In the applied cryptography community, Ellison's ceremonies have been used as a basis for modelling authentication ceremonies involving humans [5]. In the authentication ceremony described by Brainard et al., a human who still has their primary authentication details intact, the *helper*, vouches for another personally known human who has lost their authentication details (the *asker*). This vouching process, an extra factor in identification of the asker, allows emergency authentication details to be provided.

There is a large body of work on such topics as phishing on the internet, and social engineering in general [6,10]. This reflects the common understanding that many security decisions are based on trust, such as trust in a brand, rather

[1] `http://weis2010.econinfosec.org`
[2] `http://www.cl.cam.ac.uk/~rja14/shb10/`

than the mathematical assurances of a correctly executed protocol. For this reason, ceremony analysis provides a more complete understanding of the issues surrounding the use of a protocol by a human, than protocol analysis alone.

1.2 Contribution

We reinterpret recently identified security flaws in the context of ceremonies, and use this information to establish a list of typical flaws in ceremonies. We apply the knowledge learned from analysing previously identified security flaws to analyse a protocol including a human which has been proven secure. In doing so, we show that ceremony analysis is powerful, in that it can capture and describe all of the known issues investigated, and highlight flaws in a protocol proven secure. However, the process yielded the knowledge that ceremony analysis is analysis of one particular "use case" of a (set of) protocol(s). This knowledge leads to the realization of a limitation of ceremony analysis, which is that if the context of the set of protocols is changed then what was secure may no longer be secure (a different context, even for the same set of protocols, is a different ceremony).

1.3 Outline

In the next section we give an overview of ceremonies and reinvestigate the Hypertext Transfer Protocol Secure (HTTPS) ceremony from Ellison. After this introduction to ceremonies, the analysis of the Opera Mini ceremony is shown. We analyse three ceremony investigations, including an investigation of an EMV (Europay, MasterCard and VISA) ceremony not shown due to space constraints, and discuss the findings. We then use the lessons learned from the analysis of these known flaws to analyse a protocol which has been proven secure using an adversarial security model. Finally, we outline the strengths and weaknesses of ceremony analysis.

2 Ceremonies

Ellison wrote about security ceremonies in 2007 [7]. In this paper, he attributed the name *ceremony* as being coined for this purpose by Jesse Walker. Ellison provided several central ideas in a network security context, which can be directly applied to cryptographic protocols in general. The properties of a security ceremony that we distil from Ellison's work are as follows:

- a ceremony is a superset of protocols;
- there is nothing out-of-band; and
- humans, when part of the ceremony, are explicitly included.

2.1 Ceremonies Example: HTTPS with MITM Attack

HTTPS is a protocol used on the internet to provide confidentiality and integrity to messages between two parties. An example HTTPS ceremony derived from Ellison's paper is shown in Figure 1. This ceremony has a number of parts, between multiple "nodes" or parties. First, on the right hand side of Figure 1,

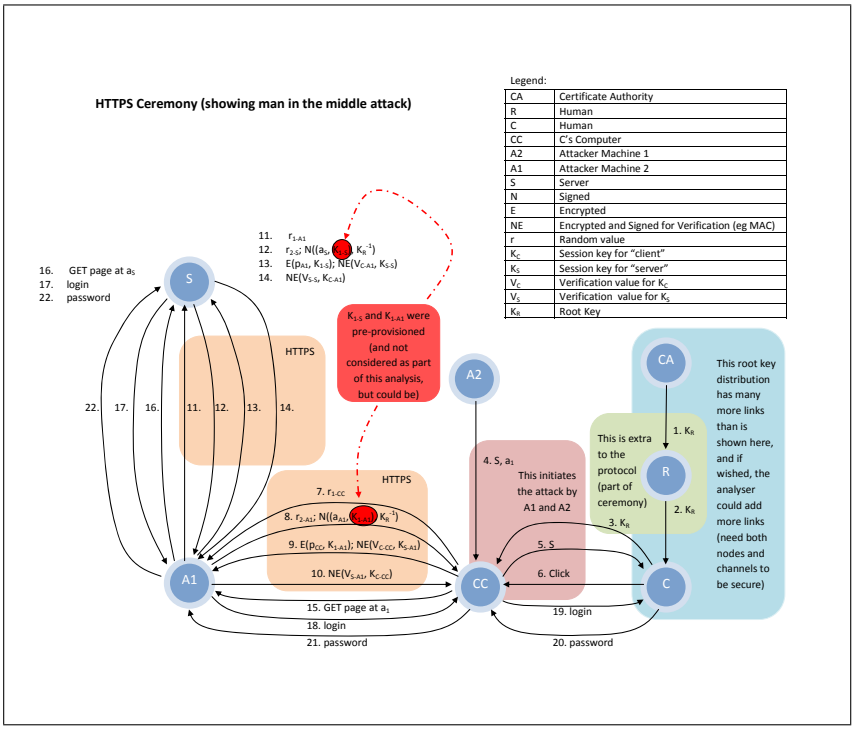


Fig. 1. HTTPS Ceremony, of Ellison (2007)

is the root key distribution part of the ceremony. The nodes in this key distribution process have been denoted by CA, R and C. Here the certificate authority is represented by CA, and R represents the registration authority which involves a number of human steps between the CA and the human party C. The human C will use the key from the CA on C's computer CC. The messages for placing the key on C's computer CC are shown in messages 1 to 3. Notice that there is no time scale on the ceremony.

The attack is shown between the user C, and the user's computer CC and the server S, in messages 4 to 6. The attack is carried out via two adversaries, $A1$ and $A2$. At some time after the user's computer CC is set up ready to take part in HTTPS, adversary $A2$ sends a name (server S's name) and an address (adversary $A1$'s address) to the computer CC. User C decides whether or not to proceed to the server based on the server's name alone, because the software running on CC does not present both the name and address to user C, only the name.

From here, the ceremony proceeds as expected through messages 7 to 22, and hence the attack. User C's computer, CC, securely connects to adversary $A1$ (messages 7 to 10) using HTTPS, adversary $A1$ securely connects to server S (messages 11 to 14) using HTTPS, and then the adversary $A1$ faithfully relays communication between the user's computer CC and the server S. Specifically

$A1$ passes on the login and password information, which adversary $A1$ now has in plaintext form for the future (note the decryption and re-encryption between messages 21 and 22 for the password, and similarly for the login). After message 22, adversary $A1$ is securely logged into server S, and is free to proceed as desired.

Ellison's example ceremony presumes that only the name of the target, and not the target's web address, is passed on to the human through the web browser in message 5, for the human to make their decision on. If this is the case, then this is clearly an issue that will result in the security of the ceremony being compromised. Some readers may suggest that this should not be the case any longer, due to such advances as extended certificates which have been introduced since 2007 (`http://www.cabforum.org/`). However, in a recent study by the authors which asked the participants to log their web usage security decisions for a week, not one participant based any of their security decisions in a week of web use on any of the information made available by the extended certificate enhancements [15]. Also, extended certificates are not yet mandated for use in HTTPS. Hence the issue remains current. Further, even if the address, as well as the name, is displayed to the user to base their security decision on, Ellison asks whether the human user will be provisioned ahead of time with the association between the address of the server and the name of the server, and the correctness of the name [7].

The above means that, in the ceremony shown in Figure 1, the user (C) believes that their computer (CC) is securely connected to the server (S). Indeed, CC is securely connected to *something*, just not the *intended* server. The point is that the HTTPS protocol is not broken, there are successful usages of the protocol between CC and $A1$, and between $A1$ and S. But the security ceremony, which includes the HTTPS protocol, is fatally flawed.

3 Opera Mini Ceremony

Opera Software ASA is a company which develops a suite of multi-platform web browsing software programs (`http://www.opera.com/company/`). Opera has had the greatest market share of any mobile web browser in the world, for the last 12 months [3]. There are different versions of Opera web browsers for different purposes. The three main variations of the browser being:

- standard Opera for PC/Mac
- Opera Mini for mobile telephones
- Opera Mobile for devices such as PDAs

3.1 Opera Mini Design

Opera Mini is the version for devices such as mobile telephones, which have restricted computing power and resources. Opera Mini has no full rendering engine on the device (`http://www.opera.com/mobile/specs/`). Instead, Opera has proprietary servers which handle the internet requests made on the mobile.

[3] `http://gs.statcounter.com/#mobile_browser-ww-monthly-200911-201010`

This process of sending requests to the internet via a server which handles the rendering and compresses the data before sending the resulting page back to the mobile telephone, has benefits both in a reduction of the computing power required on the device, and also reduced bandwidth requirements to the device which is running Opera Mini. The issue from a security point of view is that there is no *end-to-end* security. The requests from the mobile telephone to Opera's server are encrypted using Opera's proprietary encryption, but the messages are decrypted from Opera's proprietary encryption at the Opera server, and then the data is re-encrypted using standard HTTPS and the certificate of the actual target website (`http://www.opera.com/mobile/help/faq/#security`). As the Opera Mini FAQ on security reads:

> "To be able to do this translation, the Opera Mini server needs to have access to the unencrypted version of the webpage. Therefore no end-to-end encryption between the client and the remote web server is possible. If you need full end-to-end encryption, you should use a full web browser..."
> (`http://www.opera.com/mobile/help/faq/#security`)

3.2 Opera Mini Ceremony Analysis

Opera mini's use in a mobile phone is a quintessential security ceremony. There is one protocol between Opera's server and the internet, another protocol between the mobile telephone and Opera's server, and finally there is a human user making security decisions based on what they see on the browser on their mobile telephone.

Of particular interest in the Opera Mini ceremony is the use of standard icons to indicate security to the user. In, for example, Internet Explorer, which almost one in two desktop users currently use worldwide[4], the use of the *padlock* symbol means that the connection between the user and the website the user is interacting with is secure via use of HTTPS. By secure, we mean that confidentiality and integrity are assured such that no computer on the path between the user and the website can decrypt any of the information or change the message that is sent by the user or the website. The padlock icon is used similarly in all other major browsers.

However, as shown in Figure 2, Opera Mini displays a padlock symbol (top right of picture) when there is not end-to-end security. This means that Opera Mini users, who know what the padlock symbol means in other browsers, are led to believe that they have a confidential connection to the website they are viewing, when they do not.

Figure 3 describes the Opera Mini ceremony. The ceremony begins with the user of a mobile telephone typing the address of their bank's website into the Opera Mini web browser (message 1). A process similar to HTTPS then occurs between the mobile telephone and Opera's Server (approximated by messages 2 to 5). As Opera ASA states:

[4] `http://gs.statcounter.com/#browser-ww-monthly-200907-201008`

Fig. 2. Opera Mini Secure Connection (`http://www.opera.com/mobile/demo/` viewing NAB's secure logon page)

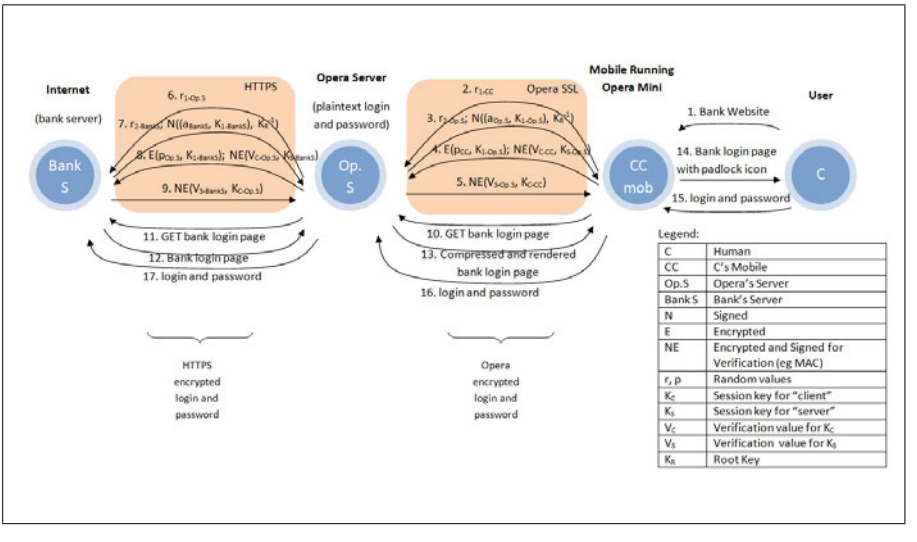


Fig. 3. Opera Mini Ceremony

The communication is protected by 256-bit RC4 and the key exchange is done by 1280-bit RSA. All hashes are created using SHA-256. These are the algorithms used by most SSL sites today. (`http://www.opera.com/mobile/help/faq/#security`)

A HTTPS connection is also formed between Opera's server and the bank's server (messages 6 to 9). Once this is complete, the request for the page is passed through to the bank (messages 10 and 11), and the bank replies with its customer login page (message 12). The Opera server renders this page, and sends the compressed output to the user's mobile telephone device (message 13). On the mobile telephone, Opera Mini then displays the webpage, including the padlock symbol (message 14). The user sees the padlock symbol, and chooses whether to input their login information and password. If the user does enter their login and password (message 15), then this is sent back to the bank's server via the Opera encrypted channel (message 16), decrypted at the Opera Server, and then re-encrypted and sent on to the bank's server via the HTTPS encrypted channel (message 17).

In a recent study, our research team investigated security decisions made by users in a week of standard web usage. We found that most users made the choice of whether or not to interact with websites that had direct financial interfaces, such as banks or online retail, based on whether or not the padlock symbol was shown [15]. Users presumed that a padlock meant that no one, apart from the website they were communicating with, could see their financial details and confidential information, such as login and password, in plaintext form. Opera's intimation of confidentiality by the depiction of the padlock symbol is not in keeping with Opera's statement in the Opera Mini FAQ which says "if you need full end-to-end encryption, you should use a full web browser..." (`http://www.opera.com/mobile/help/faq/#security`).

Interestingly, while the plaintext state of messages through the Opera Server clearly is a security issue and probably not realised by most Opera Mini users, the design has some security benefits. If the user trusts Opera Mini with all their communication with every party they communicate with on the internet, then this design of accessing the internet through a proxy provides essentially anonymous internet usage, as well as protection against various JavaScript-based malicious software (malware).

4 Lessons Learned

By re-investigating known security flaws from a ceremony point of view, we identified a set of common flaws. These ceremonies included the EMV ceremony described by Murdoch et al., but these were left out of this paper due to space constraints [14]. This list included:

- each individual protocol remained secure, but the critical security information was not passed from one protocol to the next;
- the information passed on to the human was inadequate for the human to have any chance of making a correct decision;
- it is clear that lessons long since learned for protocols, have not been transferred into security ceremony knowledge.

While ceremony analysis has been demonstrated to capture known flaws, and therefore is useful, the technique is not without pitfalls. The most significant

flaw is highlighted by our definition for a ceremony, stated in Section 1, which was that security ceremonies were *protocols in their context of use*. This means that, even if the underlying protocols are found to be secure for a given context, they may well not be secure in even a slightly different context, leading to the situation of requiring a new ceremony analysis for the same set of protocols in each new context.

All of the ceremonies examined have been *use cases*, the *context of use*, of the underlying protocols, and therefore the first job of the ceremony analyser is to create a list of use cases to create a rigorous security proof for. Of particular concern for the ceremony analysis technique are areas where the context of use for the protocols, for a specific ceremony, do not yet exist. Ceremony analysis will therefore, by necessity, trail behind users' use of any given system. For example, the people responsible for the security of new smart card driver licenses will only be able to analyse certain security ceremonies once users of the smart card have been interacting with (potentially previously unknown) third parties. This interaction with new third parties may be a new context, and hence a new ceremony will be created which will be able to be analysed only in retrospect. This is a significant step down from the ideals of provable security, which aims to ensure that, once a protocol is proven secure, it will be secure regardless of how it is used.

Therefore the common flaws revealed in the ceremonies analysed to date suggest these assessments which should be completed on security ceremonies prior to deployment.

- Look for protocol-like deficiencies, such as outlined in [1]. Treat each constituent protocol as a node in the ceremony, and check that nonces and identification are being passed between nodes.
- Ensure that key cryptographic information is being transferred between nodes in the ceremony.
- If the ceremony includes a protocol including a human as part of the protocol, and if the protocol comes with a proof of security, re-examine the proof of security for the assumptions that were made concerning the human.
- Examine the human's role in the ceremony. If the only way for the human to accomplish their goal is via a particular route through a security decision point, the human will take that route.
- Examine the human-factor considerations of the ceremony. These issues include how many items a human can remember (for example, web address and store name pairs, as per the HTTPS ceremony) and the prior knowledge and education required. For example, in approving the usage of a HTTPS ceremony, do humans realise that the most critical information is the address? Our recent study indicated that they did not.

5 Investigation of a Provably Secure Protocol

In 2008, Gajek et al. expanded on Bellare and Rogaway's concept of *practice oriented provable security* [3]. The significant enhancement that Gajek et al.

made to previous security models was that they proved a protocol including a human to be secure [9]. They achieved this by adding formal actions *render* and *recognise* to a security model. *Render* is the process of a web browser rendering a HTML page, based on the browser's state, and presenting that page to the user. *Recognise* is the process of a user viewing the webpage, judging if the *Human-Perceptable Authenticator* (HPA) is correct, and outputting either *true* or *false* depending on the results of that test.

The protocol that Gajek et al. proved to be secure, what they called *browser-based user-aware mutual authentication over TLS*, is a non-trivial security ceremony. In the protocol, there is a user who has a computer, a browser running on the user's computer, and the user is interacting with a server via their computer's browser. Gajek et al. take the important step of extending the definition of the underlying TLS (Transport Layer Security) protocol to include the human user. In the protocol, the user types the address of the server into their browser, the TLS HTTPS connection is created between the server and the browser, the server then sends a HPA to the user via the browser (which renders the HPA). If the user recognises the HPA, then they type in their login credentials. In this way, the server is authenticated to the user (via the HPA) and the user is authenticated to the server (via the traditional login and password technique). For the full Gajek et al protocol, see [9].

We analysed the Gajek protocol using the lessons learned via our analysis of the previously outlined security ceremonies. We make the following observations.

- The protocol begins with the human typing in the web address of the server. This immediately removes one significant source of failing by the user (web address of target incorrect), which was specifically outlined in the HTTPS Ceremony shown in section 2.1. So the question is only, "Could an adversary, who is not the server, supply the user with a HPA which will cause the user to enter their username and password?"
- Many assumptions are rolled into the browser's *render* and the human's *recognise* capabilities. For example, since it is not specified in the protocol, there is every chance that a website (and browser) designer implementing this protocol would not put in any check to ensure that the HPA (typically a picture) is fully shown *before* the user can type in their user name and password. On a slow connection, users may well type in their details prior to seeing some, or all, of the HPA. Further, even if such a check was put in place (picture fully downloaded and shown prior to displaying login details entry form), the protocol could still be broken via sending an *all black* or *all grey with a red cross in the middle* picture. Many users may view these pictures as a download or rendering fault, and still enter their user name and password.
- Another potential attack is to degrade or pixelate the picture. There will be storage space and bandwidth decisions made concerning the file format, size, and resolution of the HPA, by at least all three of the owners of the server, the webpage developer, and the web browser developer. As written, these decisions are left to the individual developers with no necessity for

a common technique. The essential message, from both this and the prior point, is the need for protocol developers to include in their protocol design, and hence protocol proof, the specification of the critical elements of the designs of the interface to the human.
- How does the user know that *this is the protocol*? The user does not know the algorithm, does not know that suddenly they should be waiting for a HPA. This suggests that there is no need to attack this protocol at all, and the adversary should create a different protocol. Therefore, once this issue is realised, as part of a security *ceremony* potential solutions such as side-channel instructions to the user about the protocol may be necessary.

6 Conclusion

We have shown that security flaws in complex systems of protocols, with human interaction, can be analysed using security ceremonies. The analysis of the EMV smart card ceremony (omitted due to space constraints) and the Opera Mini ceremony, followed by the analysis of the TLS protocol which had been proven secure for human use, shows that a ceremony analysis is capable of capturing a greater range of security flaws than protocol analysis alone.

In the process of analysing these ceremonies, we have constructed an approach for analysing further security ceremonies. We also highlight the role that the designer plays in ensuring that the ceremony is secure. This role necessitates a grounding in security considerations, and similarly that creators of protocols are aware of typical design considerations at the human-computer interface.

Finally, the realisation that security ceremonies are essentially *use cases* of the underlying protocols, warns against the presumption that a protocol shown secure in one ceremony will mean that the same protocol is secure in another ceremony. The development of a list of use cases for the protocol, or device such as a smart card, becomes critical, as is the use standardized protocols that either are provably secure or have been rigorously scrutinized. This work may be similar to the construction of a safety case for mission critical systems.

Acknowledgments. The authors acknowledge and appreciate the suggestions by Jason Reid and Douglas Stebila as to appropriate real-world groups of protocols we could conduct ceremony analysis on. The authors also appreciate the quality and differing viewpoints exhibited by the blind reviewers, which have directly lead to improvements in this paper.

References

1. Abadi, M., Needham, R.M.: Prudent engineering practice for cryptographic protocols. IEEE Trans. Software Eng. 22(1), 6–15 (1996)
2. Bella, G.: Formal correctness of security protocols. Springer, Heidelberg (2007)
3. Bellare, M.: Practice-oriented provable-security. In: Damgård, I. (ed.) EEF School 1998. LNCS, vol. 1561, pp. 1–15. Springer, Heidelberg (1999)

4. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
5. Brainard, J.G., Juels, A., Rivest, R.L., Szydlo, M., Yung, M.: Fourth-factor authentication: somebody you know. In: ACM Conference on Computer and Communications Security, pp. 168–178. ACM, New York (2006)
6. Dhamija, R., Tygar, J., Hearst, M.: Why phishing works. In: Proceedings of the SIGCHI conference on Human Factors in computing systems, p. 590. ACM, New York (2006)
7. Ellison, C.: Ceremony Design and Analysis. Cryptology ePrint Archive, Report 2007/399 (2007), http://eprint.iacr.org/
8. Ellison, C., Dohrmann, S.: Public-key support for group collaboration. ACM Trans. Inf. Syst. Secur. 6(4), 547–565 (2003)
9. Gajek, S., Manulis, M., Sadeghi, A.R., Schwenk, J.: Provably Secure Browser-Based User-Aware Mutual Authentication over TLS. In: Abe, M., Gligor, V.D. (eds.) ASIACCS, pp. 300–311. ACM, New York (2008)
10. Herzberg, A.: Why Johnny can't surf (safely)? Attacks and defenses for web users. Computers & Security 28(1-2), 63–71 (2009)
11. Karlof, C., Tygar, J.D., Wagner, D.: Conditioned-safe ceremonies and a user study of an application to web authentication. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2009. The Internet Society, San Diego (2009)
12. Martina, J., Carlos, M.: Why should we analyze security ceremonies. In: Applications of Logic in Computer Security. In: The 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (2008)
13. Martina, J.E., de Souza, T.C.S., Custodio, R.F.: Ceremonies Formal Analysis in PKI's Context. In: CSE 2009: Proceedings of the 2009 International Conference on Computational Science and Engineering, pp. 392–398. IEEE Computer Society, Washington, DC, USA (2009)
14. Murdoch, S.J., Drimer, S., Anderson, R.J., Bond, M.: Chip and pin is broken. In: IEEE Symposium on Security and Privacy, pp. 433–446. IEEE Computer Society, Los Alamitos (2010)
15. Radke, K., Boyd, C., Brereton, M., Nieto, J.G.: How HCI Design Influences Web Security Decisions. In: OzCHI. ACM, New York (2010)
16. Ruksenas, R., Curzon, P., Blandford, A.: Detecting cognitive causes of confidentiality leaks. Electr. Notes Theor. Comput. Sci. 183, 21–38 (2007)
17. Ruksenas, R., Curzon, P., Blandford, A.: Modelling and analysing cognitive causes of security breaches. ISSE 4(2), 143–160 (2008)
18. Shostack, A., Stewart, A.: The New School of Information Security. Addison-Wesley Professional, Upper Saddle River (2008)

Preventing Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes

Reto Koenig[1,2], Rolf Haenni[1], and Stephan Fischli[1]

[1] Bern University of Applied Sciences, CH-2501 Biel, Switzerland
{rolf.haenni,stephan.fischli}@bfh.ch
[2] University of Fribourg, CH-1700 Fribourg, Switzerland
reto.koenig@unifr.ch

Abstract. This paper addresses the board flooding problem of Juels et al.'s coercion-resistant electronic voting scheme. A key property of this scheme is the possibility of casting invalid votes to the public board, which are indistinguishable from proper votes. Exactly this possibility is crucial for making the scheme coercion-resistant, but it also opens doors for flooding the board with an enormous amount of invalid votes, therefore spoiling the efficiency of the tallying process. To prevent such attacks, we present an adaption of the scheme in which each voter receives —in addition to the proper credential—some dummy credentials from the election registrars. Dummy credentials may be used to deceive possible coercers. The list of all dummy credentials is published along with the electoral register. Based on the electoral register and the list of dummy credentials, the system is now capable of making a distinction between invalid votes generated from dummy credentials and invalid votes generated from fake credentials. While the former are kept until the tallying phase, the latter are immediately rejected by the public board. If the public board additionally rejects all incoming duplicate votes, then its maximum size is bounded by the total number of issued credentials. This guarantees an efficient linear-time tallying phase even in case of a massive board flooding attack with a very large number of invalid votes. Although the solution presented in this paper does not yet entirely rule out vote selling or coercion, it makes it at least unbearable for the vast majority of voters.

1 Introduction

One of the most challenging problems in remote electronic voting is the design of a system that prevents voters from selling their votes or from being coerced. The first scheme that is resistant against both the selling of votes and the coercion of voters has been proposed by Juels, Catalano, and Jakobsson in [7]. To achieve *coercion-resistance* (which implies mere *receipt-freeness*), the so-called "JCJ-scheme" uses an anonymous authentication mechanism to guarantee that the identities of the voters remain hidden during the whole voting and tallying process. The anonymous authentication mechanism requires that during the registration phase each voter receives a *secret credential* over an untappable

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 116–127, 2011.

channel. The knowledge of the secret credential allows the voter then to post an encrypted vote anonymously to the public board, such that its inclusion in the final tally is guaranteed. It is also possible to post invalid votes based on *fake credentials*, but those will be filtered out later during the tallying phase. Since board entries created from proper credentials are indistinguishable from those created from fake credentials, it is always possible to lie about the secret credential or to supply a coercer with a fake credential. The vote buyer or coercer will then see the posted invalid vote on the public board, but at this early stage of the protocol, there is no way to tell whether a particular board entry will be included in the final tally or not. This is the principal mechanism that renders the JCJ-scheme coercion-resistant.

The JCJ-scheme is the point of departure of most advanced protocols for remote electronic voting today dealing with coercion-resistance, but the protocol as presented in [7] has at least two major open problems.[1] The first problem is the quadratic running time of the tallying process, where duplicate and invalid votes need to be eliminated. Detecting duplicate votes requires so-called *plaintext equivalence tests* (PET) [6] for every pair of votes cast, and detecting invalid votes requires each vote cast to be checked against the public electoral register, thus making the scheme quite inefficient for large scale elections. The *Civitas system* [3], an implementation of the JCJ-scheme, weakens this problem by breaking up the electoral register into various independent blocks of a given fixed size. Several other improvements based on hash tables were proposed by Smith, Weber, and others [8,13,14,16,17], but they have been shown to be vulnerable to Pfitzmann's attack against anonymous channels [12]. More recent developments in this direction are based on group signatures [1,2] or fake votes generated by the talliers [15].

The second major problem of the JCJ-scheme results from the aforementioned possibility of posting invalid votes based on fake credentials to the public board. Exactly this possibility is crucial for making the scheme coercion-resistant, but it also opens doors for flooding the public board with an enormous amount of invalid votes. Because invalid votes are indistinguishable from proper votes from the perspective of the public board, there are no direct counter-measures against such types of attack, i.e., as long as the incoming votes cast are well-formed and comply with the protocol, the public board needs to treat them all in the exactly same manner. A massive application-level flooding attack of that kind may therefore both jeopardize the availability of the public board and spoil the efficiency of the tallying process. To our best knowledge, no practical solution to this problem has yet been proposed in the literature. The problem itself seems to be intrinsic to the chosen approach.

In this paper, we propose an extension of the JCJ-scheme that addresses both aforementioned problems. The key idea is to equip the public board with a stronger filter on what is an acceptable vote cast. For this, the voters receive during the registration phase some *dummy credentials* (in addition to the secret credential), which may then be used to mislead potential vote buyers or coercers.

[1] Further major and minor problems of the JCJ-scheme are discussed in [9,14].

Invalid votes generated from these dummy credentials will be accepted by the public board (and filtered out later), but invalid votes from fake credentials will be rejected immediately. As we will see, enhancing the JCJ-scheme in such a way has a number of potential pitfalls. These pitfalls will be discussed and possible solutions will be presented.

The major benefit of our method results from the public board's ability to separate invalid votes created by fake credentials from those created by dummy credentials. Let n denote the number of voters, m the number of issued dummy credentials, and s the number of votes cast using fake credentials. Note that n and m are fixed during the registration phase, whereas s is unbounded (and possibly orders of magnitude larger than $n + m$). If we further assume that the public board is also capable of eliminating duplicate votes, we can introduce an upper limit $n + m$ for the size of the public board.

Another important benefit of our approach is the fact that the known attacks against the linear-time improvements proposed by Smith [14] and Weber [16,17] are no longer possible. The reintroduction of these improvements allows the elimination of all types of invalid votes (duplicate, fake, and dummy) in linear time, which reduces the total running time of the original JCJ tallying phase from $O(n^2 + s^2)$ to $O(n + m)$. If furthermore $m = d \cdot n$ for some constant $d > 0$ (the average number of dummy credentials issued per voter), then the tallying phase even runs in $O(n)$ time.

Unfortunately, this unprecedented leap in performance and robustness has some negative effect with respect to perfect coercion-resistance. It is possible to minimize this effect to a small subset of unfortunate voters, which receive the minimum amount of dummy credentials, but some residual affliction will remain. The same holds true for vote buying. We will discuss this topic and see how to further minimize this problem.

The paper is organized in the following way. In Section 2, we give a short overview of the original JCJ-scheme and discuss its properties and problems. The proposed solution for the board flooding problem and the corresponding extension of the JCJ-scheme is discussed in Section 3. We first exhibit the general idea, then give a semi-formal sketch of the adapted protocol, and finally discuss some of the above-mentioned pitfalls. Section 4 concludes the paper.

2 Coercion-Resistant E-Voting

The goal of the scheme proposed by Jules, Catalano, and Jakobsson in [7] is to make remote electronic voting resistant against all sorts of coercion. Coercion-resistance is defined as a stronger form of privacy. While privacy is defined in terms of an adversary that cannot interact with voters during the election process, it is assumed that a coersive adversary may interact with voters at any time. Thus an election scheme is called *private*, if the adversary cannot guess the vote of any voter better than an adversarial algorithm whose only input is the final tally, and the scheme is called *coercion-resistant*, if the adversary can be deceived into thinking that a coerced voter has behaved as instructed. Such

a scheme thus prevents voters from selling their votes or from being coerced in various ways, e.g. to vote in a particular way, to vote at random, to abstain from voting, or even to divulge the private keying material [7].

The JCJ-scheme is the first electronic voting protocol that offers full coercion-resistance under minimal assumptions. While many other protocols assume the existence of an untappable channel during the voting phase to offer mere receipt-freeness, an untappable channel is only required during the registration phase of the JCJ-scheme. Note that this assumption is realistic, because the registration process often requires the voters to visit the registration office in person. We will now briefly describe the JCJ-scheme in a semi-formal way. The main entities in the protocol beside the voters are the following:

Registrars. They issue the secret credential to voter V_i and pronounce corresponding encryptions publicly to the system. A threshold encryption system guarantees that the secret credential is only known to the voter and that the protocol is safe even if a minority of the registrars is corrupted or under attack.

Tallying Authorities. They are responsible for processing the votes cast, jointly decrypting and counting the votes, and publishing the final tally. Again, a threshold encryption systems guarantees the safety of the protocol even if a minority of the tallying authorities is corrupted or under attack.

The votes cast are published on an append-only *public board*. Its task is to accept and publish every well-formed vote cast that complies with the protocol. To guarantee the integrity and availability of the board, it may be replicated in such a way that a minority of unavailable or corrupted board servers does not prevent its functioning properly as a whole [5,11].

The whole voting protocol is divided into three major phases, during which the voters are authenticated anonymously. The protocol uses numerous cryptographic primitives such as encryption, digital signatures, non-interactive zero-knowledge proofs of knowledge, plaintext equivalence tests, re-encryption mix-nets, anonymous channels, etc. A first overview of the protocol is given in Figure 1.

Registration. Each voter V_i, $1 \leq i \leq n$, receives a secret credential σ_i jointly generated by the registrars. This credential constitutes a proof of eligibility, which is used in the voting phase to cast the vote. Additionally, an encryption $S_i = \mathrm{Enc}_\varepsilon(\sigma_i, \gamma_i)$ of σ_i with randomness γ_i is appended to the (digitally signed) electoral register on the public board. ε denotes the tallying authorities' common public key. The protocol assumes the majority of the registrars to be trustworthy and the channel between the registrars and V_i to be untappable.

Voting. The voters cast their candidate selection $c_i \in \mathcal{C}$ via an anonymous channel to the public board. The message posted to the board consists of $A_i = \mathrm{Enc}_\varepsilon(\sigma_i, \alpha_i)$ and $B_i = \mathrm{Enc}_\varepsilon(c_i, \beta_i)$ along with corresponding zero-knowledge proofs of knowledge of σ_i and c_i. It is important that the candidate set $\mathcal{C}$ is finite and that an additional disjunctive proof that c_i represents a valid candidate choice is provided. This proof is needed to prevent the construction of receipts based on invalid candidate choices.

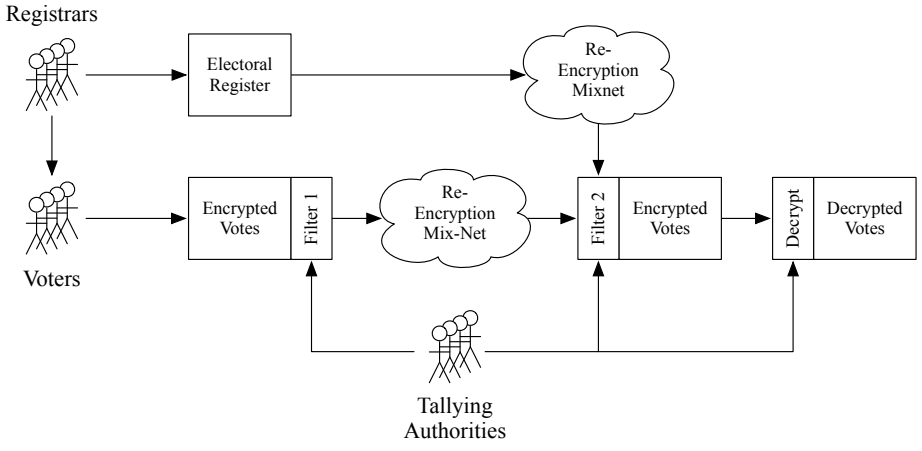


Fig. 1. Overview of the original JCJ-scheme: the first filter eliminates votes with invalid proofs and duplicate votes from the public board, while the second filter checks the votes cast against the electoral register (and thus eliminates votes created from fake credentials)

Tallying. The tallying authorities check the proofs included in the votes cast and jointly perform pairwise PETs on the encrypted credentials to eliminate duplicates. The resulting adjusted list of votes cast is shuffled in a verifiable re-encryption mix-net to anonymize the votes and credentials included. Respective proofs of correct shuffling are published on the public board. Another verifiable re-encryption mix-net is applied to the electoral register, which finally allows the tallying authorities to jointly check the validity of the encrypted credentials involved in the votes cast (without decrypting them). Votes accompanied with fake credentials are discarded. The resulting adjusted list of proper votes is decrypted and tallied.

What makes this particular system coercion-resistant is the fact, that any posted entry to the public board is accepted if it is well-formed and complies with the protocol. It must thus consist of a valid candidate selection and some credential encrypted by the tallying authorities' common public key (together with corresponding proofs of knowledge). But the credential must not necessarily be a proper credential issued by the registrars and thus constituting a proof of eligibility, it simply needs to have the format of a proper secret credential. This enables the voter to deceive potential coercers with a *fake credential*, simply by choosing one at random. Votes accompanied with such fake credentials are discarded during the tallying phase. The two mix-nets involved in the tallying phase guarantee that no voter can prove to a third party whether a particular vote cast has been discarded before tallying or not. This feature makes the system resistant against selling votes or coercing voters.

Scheme by Smith and Weber [14,16,17]. Instead of applying PETs on all pairs of distinct votes for removing duplicates, both Smith and Weber in

essence suggest computing and decrypting $A_i^z = \mathrm{Enc}_\varepsilon(\sigma_i^z, \alpha_i^z)$, where $z \in \mathbb{Z}_q$ is a random value shared among the talliers. The resulting *blinded credentials* σ_i^z are stored in a hash table for collision detection in linear time (clearly, $\sigma_i = \sigma_j$, iff $\sigma_i^z = \sigma_j^z$). Both authors propose using the same procedure for eliminating votes created from fake credentials, but since the same exponent z is used across all ciphertexts A_i, the coercer gets an attack strategy to identify whether a vote with known σ_i is counted, namely by posting two votes, one that includes an encryption of σ_i and one an encryption of σ_i^2 [1,3,12]. Note that this attack is not applicable to the mere removal of duplicates.

3 Preventing Board Flooding Attacks

In this section, we describe a way of modifying the JCJ-scheme to become resistant against board flooding attacks and to allow a linear-time tallying phase. As this implies several major and minor changes to the JCJ-scheme throughout various parts of the scheme due to different reasons, we uncover them step by step. A discussion of some important related questions follows in the second part of this section.

3.1 The Modified JCJ-Scheme

To protect the public board against application-level flooding attacks, it needs to be equipped with a stronger filter on what to accept. The main idea of our approach is to accept only votes cast from legitimate voters. Since it is crucial for the original JCJ-scheme to accept any vote cast, even those accompanied with a fake credential, it seems to be impossible in the first place to make such a distinction between legitimate and non-legitimate voters. But by introducing a third category of credentials, so-called *dummy credentials*, which are distributed to the voters during the registration phase (together with the proper secret credential), it is possible to reject all votes accompanied by fake credentials right from the beginning. Thus the idea is that the dummy credentials take over the role of deceiving potential vote buyers or coercers. This means that during the vote casting phase, they need to be treated in the same way as the secret credentials, whereas fake credentials are immediately rejected. In other words, voters are equipped with several access keys for posting votes to the public board, but only one of them is a key to include votes in the final tally. A first overview of the extended protocol is given in Figure 2.

More formally, let $\{\tau_{ij} : 1 \leq j \leq d_i\}$ be the set of dummy credentials for voter V_i (note that that d_i might be different for every voter, see Subsection 3.2). They are generated jointly by the registrars during the registration phase, together with V_i's secret credential σ_i. Corresponding encryptions $T_{ij} = \mathrm{Enc}_\varepsilon(\tau_{ij}, \gamma_{ij})$ with randomness γ_{ij} are published on the public board together with $S_i = \mathrm{Enc}_\varepsilon(\sigma_i, \gamma_i)$. The public board thus contains two separate lists of encrypted credentials: the original electoral register $\mathcal{S} = \{S_i : 1 \leq i \leq n\}$ and the new set $\mathcal{T} = \{T_{ij} : 1 \leq i \leq n, 1 \leq j \leq d_i\}$ of dummy credentials. With $\mathcal{ST} = \mathcal{S} \cup \mathcal{T}$ we

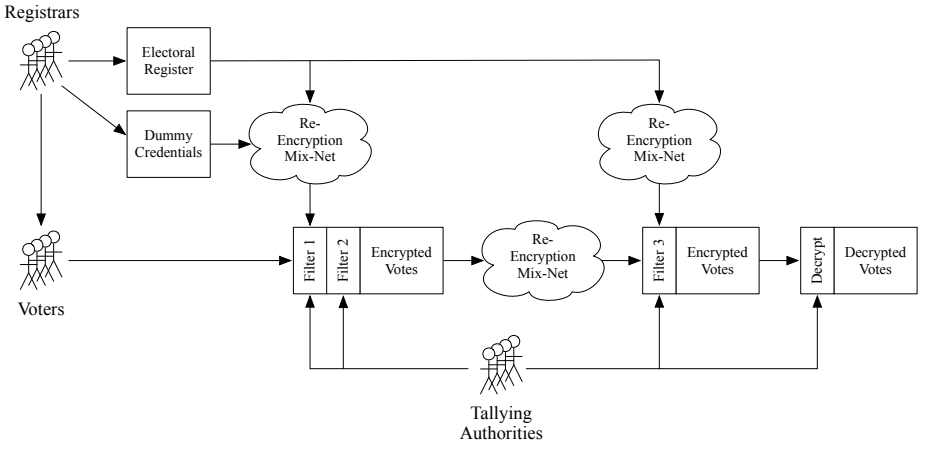


Fig. 2. Overview of the extended JCJ-scheme: the first filter discards votes created from fake credentials, the second filter removes duplicates, and the third filter checks the votes against the electoral register (and thus eliminates votes created from dummy credentials). The final list of proper votes is decrypted and counted.

denote the complete set of encrypted credentials. Furthermore, we denote the number of all issued dummy credentials by $m = |\mathcal{T}| = \sum_{i=1}^{n} d_i$, which implies that a total of $|\mathcal{ST}| = n + m$ credentials have been issued in all. Those are the ones that will be accepted by the public board during the vote casting phase.

To detect fake credentials for filtering out corresponding invalid votes, the public board needs to check for each incoming vote cast whether the included encrypted credential matches with one of the entries in the list $\mathcal{ST}$. We can safely apply Smith's and Weber's linear-time scheme here, because all votes based on arbitrarily fake credentials are already dropped at this early stage. Note that to install this first filter, we require the help of the talliers already during the voting phase. Compared to the original JCJ-scheme, this is a true disadvantage of our approach, but it is the key for restricting the size of the public board to an upper limit. To do so, the detection and removal of duplicate votes needs to be performed simultaneously, again by applying safely Smith's and Weber's scheme and with the help of the tallying authorities. In Figure 2, these tasks of the public board are called "Filter 1" and "Filter 2", respectively. Note that $\mathcal{ST}$ needs to be shuffled in a verifiable re-encryption mix-net, similar to the shuffling of $\mathcal{S}$ in the original JCJ-scheme. This is important for disguising the links between the voters and their entries in $\mathcal{ST}$.

The rest of the tallying phase is similar to the original JCJ-scheme, except that the elimination of duplicate votes has already been conducted. Therefore both, the list of encrypted votes registered on the public board and the list $\mathcal{S}$ of encrypted secret credentials, are shuffled in corresponding re-encryption mix-nets. Respective proofs of correct shuffling are published. The output of the two mix-nets are then used to separate the valid votes from those generated by dummy credentials, again by applying safely Smith's and Weber's linear-time

scheme. In Figure 2, this task is called "Filter 3". At the end, the adjusted list of encrypted votes is jointly decrypted and tallied by the tallying authorities.

3.2 Discussion

The above description of the adapted JCJ-scheme outlines the general ideas of our approach. The modifications raise several important questions. Some of them will be discussed below.

How many dummy credentials are needed? To answer this question, suppose first that each voter receives exactly $d \geq 1$ dummy credentials from the registrars, i.e., let $d_i = d$ for all $1 \leq i \leq n$. Each voter would then have d extra credentials to deceive potential vote buyers or coercers. The problem of such a scenario is that the secret credential could only be withhold as long as not all d dummy credentials are "expended". A coercer could thus force the voter to release all $d + 1$ credentials and use them to cast $d + 1$ identical votes. If all $d + 1$ votes cast appear on the public board, it follows that one of them (the one that includes the proper credential σ_i) will be included in the final tally. Otherwise, if some of the votes cast do not appear on the public board, then the coercer knows that the voter was lying about some of the credentials. Using a similar line of reasoning, votes could be sold by passing all $d + 1$ credential to a vote buyer. Therefore, a constant number of dummy credentials clearly ruins the coercion-resistance property of the scheme.

The above argument leads to the conclusion that the registrars have to generate a varying number of dummy credentials for each voter. Suppose that V_i receives $d_i \in_R \{1, \ldots, d\}$ dummy credentials, i.e., d_i is chosen at random between 1 and a fixed upper limit d. If the scheme guarantees that d_i is unknown to potential coercers (which includes the registrars and the tallying authorities), then V_i may lie about d_i, for example by passing all d_i (or less) dummy credentials to the coercer and by claiming that the secret credential is included in that list. This argument works for every V_i with $d_i > 1$, but unfortunately not for those with $d_i = 1$. Under coercion, such (unfortunate) voters could only give away a single dummy credential, but they could not claim it to be the secret credential. Note that this problem does not disappear by increasing the lower limit of the interval $\{1, \ldots, d\}$ to some value $c < d$ or by decreasing it to 0. Even worse, a similar problem exists for the upper bound d, because voters with $d_i = d$ dummy credentials could sell their votes by simply handing over all $d + 1$ credentials to the vote buyer (as in the case of a constant number of dummy credentials). This problem could be solved by not imposing an upper limit to the interval, but this brings up new problems of practicability in cases where d_i becomes very large.

As an answer to the above question, we suggest here that d_i, the number of dummy credentials for voter V_i, is determined according to some non-uniform probability distribution over sets $\mathbb{N}^d = \{1, \ldots, d\}$ or $\mathbb{N} = \{1, \ldots, \infty\}$ of natural numbers.[2] The most natural candidate distribution with an upper limit d is

[2] We explicitly exclude the borderline case $d_i = 0$, because it would completely disallow V_i to deceive a passive coercer who does nothing but directly observing V_i's vote casting process (*shoulder surfing* attack).

a *binomial distribution* $\mathcal{B}(d,p)$ with shape parameters d and p. The idea is to choose d and p such that only a very small fraction of voters get the minimum number ($d_i = 1$) or the maximum number ($d_i = d$) of dummy credentials. This is the case if the variance of the distribution is relatively small compared to d. In this way, we cannot entirely rule out vote selling or coercion, but we can at least make it unbearable for the vast majority of voters.

The most natural candidate distribution with no upper limit is a *normal distribution* $\mathcal{N}(\mu,\sigma^2)$ with some reasonable values for the mean and the variance. Since normal distributions are density functions defined over $\mathbb{R}$, they need to be applied in some discretized manner over $\mathbb{N}$. Many other distributions are possible, but a more exhaustive discussion of this questions is beyond the scope of this paper.

How do the registrars generate a random number of dummy credentials? The naïve approach for the registrars to generate a random number of dummy credentials for voter V_i is to jointly apply the chosen probability distribution to determine d_i and to generate each of the d_i dummy credentials using the same distributed procedure as for the secret credential σ_i. The problem of this simple approach is that d_i is not a secret of V_i alone, i.e., V_i cannot lie about it towards potential voter buyers or coercers if they collude with one of the registrars.

As a solution to this problem, we suggest to split up the group of registrars into r sub-groups. Each of these sub-groups is then responsible for generating roughly d_i/r dummy credentials, but without informing the other groups about the exact number. To do so, we need to decompose the chosen probability function into a sum of r probability functions with parameters adapted accordingly. In the case of a binomial distribution $\mathcal{B}(d,p)$, for example, each sub-group may simply use the distribution $\mathcal{B}(d/r,p)$ to determine their numbers independently, because $\mathcal{B}(d,p) = r{\cdot}\mathcal{B}(d/r,p)$. Similarly, a normal distribution $\mathcal{N}(\mu,\sigma^2)$ can be split up into a sum of r normal distributions $\mathcal{N}(\mu/r,\sigma^2/r^2)$, because normal distributions are closed under linear combination. Note that we need to assume a majority of each sub-group to be trustworthy.

How should the public board store the encrypted dummy credentials? In the original JCJ-scheme, the encrypted credentials S_i are published on the public board together with the plaintext identities of the voters. The list $\mathcal{S}$ plays thus the role of a *electoral register*, which can be inspected and verified by everybody. By doing the same with the encrypted dummy credentials, i.e., by linking each $T_{ij} \in \mathcal{T}$ publicly with V_i's identity, we would allow potential coercers or vote buyers to derive the secret number $d_i = |\{T_{ij} \in \mathcal{T}\}|$ from $\mathcal{T}$. But as already discussed above, coercion-resistance can only be guaranteed as long as d_i is V_i's secret, since otherwise V_i looses the ability to lie about it.

As a simple solution to this problem, we suggest that the set $\mathcal{T}$ of encrypted dummy credentials is published anonymously without any links to the voters. Since $\mathcal{T}$ does not serve as an electoral register, it does not necessarily need to be treated in exactly the same way as $\mathcal{S}$. However, this solution is only applicable if deleting entries from the electoral register is prohibited over multiple voting

events. Otherwise, additional mechanisms need to be introduced to delete the entries in $\mathcal{T}$ that belong to the deleted entry in $\mathcal{S}$.

What is the benefit of the modified scheme? The original JCJ-scheme has three critical time-consuming components: the elimination of duplicates, the mixing of the votes in the re-encryption mix-net (as well as the verification of the proofs produced by the mix-net), and the elimination of invalid votes (see Figure 1). The input size of each of these components depends directly on the number of *votes*, not the number of *voters*. Since costly cryptographic computations such as zero-knowledge proofs and multi-party computations are needed to perform these tasks, processing a single additional vote is expensive. Techniques that avoid the processing of votes that will not appear in the final tally are therefore inherently appealing.

Let $n = |\mathcal{S}|$ denote the number of voters (or the number of proper votes if all voters participate in the election) and s the number of duplicate or invalid votes. In the original JCJ-scheme, eliminating duplicates by performing pairwise PETs over all $n+s$ votes requires $O(n^2+s^2)$ relatively expensive steps. If no duplicates are removed (worst case), $n+s$ is the input size for both the re-encryption mix-net and the final procedure for eliminating fake votes. In the literature of verifiable mix-nets, we find techniques with proofs of linear size [4,10,18], but all of them involve relatively high constant factors. The final elimination of fake votes again requires $O(n^2+n\cdot s)$ expensive PETs. In total, the JCJ-scheme runs in $O(n^2+s^2)$ time and $O(n+s)$ space.

Our modified approach improves the overall performance of the scheme in respect of both, computation time and memory space. Note that we have the same three critical components, only arranged in a different order. If $m = \sum_{i=1}^{n} d_i = |\mathcal{T}|$ denotes the total amount of dummy credentials issued during the registration phase, we get an $O(n+m)$ upper limit for both the size of the public board and the input of the mix-net. As s may become orders of magnitudes larger than m in case of a large scale board flooding attack, this states a major improvement over the original scheme. It prevents situations where the system becomes unavailable due to a memory overflow of the public board.

The modified scheme also eliminates the need for the quadratic number of PETs for eliminating invalid votes. Here we benefit from the methods proposed by Smith [14] and Weber [16,17], which allow duplicate and fake votes to be detected in linear time. Therefore, all components involved in the tallying phase run in $O(n+m)$ time and space, which implies an overall $O(n+m)$ running time for the whole modified scheme. This is a considerable improvement over the original scheme under all possible circumstances.

What is the downside of the modified scheme? In the presented form, our scheme allows statistical attacks which may possibly influence the outcome of a voting event. For example, a vote buyer may offer a certain amount of money for each additional credential (dummy or proper) handed over by the voter. In this way, a potential vote seller gets a personal interest in handing over as many credentials as possible—including the proper one. An analogous strategy may be applied by the coercer. However, depending on the total number of dummy

credentials and the parameters of the chosen distribution function, these attacks may become very cost-intensive for both, the vote buyer and the coercer. This raises the question of finding the optimal distribution function to maximize the cost of such statistical attacks. Answering this question is beyond the scope of this paper and is left for future work.

4 Conclusion

In this paper, we have discussed the board flooding problem in the JCJ-scheme for remote electronic elections. As a solution, we propose that each voter receives a set of dummy credentials along with the proper secret credential during the registration phase. For this enhancement to work, it is important that the number of dummy credentials varies from one voter to an other, and that only few voters will get the minimum or maximum number of dummy credentials. The votes posted to the public board can then be filtered such that only votes created from proper or dummy credentials are retained. Duplicate votes are also immediately eliminated in linear time. In this way, there will never be more entries on the public board than the total number of issued (proper and dummy) credentials. The lack of such a filter leads to the board flooding problem in the JCJ-scheme.

This paper is a first step in transforming the impractical JCJ-scheme—applicable under the assumption of unrealistic computing power only—into a practical scheme. Our proposal allows the voting authorities to trade-off the efficiency of the tallying procedure against the obtained level of coercion-resistance. Future work will focus on the residual statistical vulnerability for coercion and vote buying. Currently, we are studying the possibility of obtaining additional dummy credentials on demand during the voting phase, for example by exchanging dummy credentials between voters.

Acknowledgments. Research supported by the *Hasler Foundation* (project No. 09037).

References

1. Araújo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Chaum, D., Kutylowski, M., Rivest, R.L., Ryan, P.Y.A. (eds.) FEE 2007, Frontiers of Electronic Voting, Schloss Dagstuhl, Germany, pp. 330–342 (2007)
2. Araújo, R., Ben Rajeb, N., Robbana, R., Traoré, J., Youssfi, S.: Towards practical and secure coercion-resistant electronic elections. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 278–297. Springer, Heidelberg (2010)
3. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: SP 2008, 29th IEEE Symposium on Security and Privacy, Oakland, USA, pp. 354–368 (2008)

4. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Journal of Cryptology 23(4), 546–579 (2010)
5. Heather, J., Lundin, D.: The append-only web bulletin board. In: Degano, P., Guttman, J., Martinelli, F. (eds.) FAST 2008. LNCS, vol. 5491, pp. 242–256. Springer, Heidelberg (2009)
6. 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)
7. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., De Capitani di Vimercati, S., Dingledine, R. (eds.) WPES 2005, 4th ACM Workshop on Privacy in the Electronic Society, Alexandria, USA, pp. 61–70 (2005)
8. Meister, G., Hühnlein, D., Eichholz, J., Araujo, R.: eVoting with the European citizen card. In: Brömme, A., Busch, C., Hühnlein, D. (eds.) BIOSIG 2008, Special Interest Group on Biometrics and Electronic Signatures, Darmstadt, Germany, pp. 67–78 (2008)
9. Meng, B.: A critical review of receipt-freeness and coercion-resistance. Information Technology Journal 8(7), 934–964 (2009)
10. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Samarati, P. (ed.) CCS 2001, 8th ACM Conference on Computer and Communications Security, Philadelphia, USA, pp. 116–125 (2001)
11. Peters, R.A.: A Secure Bulletin Board. Master's thesis, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, The Netherlands (2005)
12. Pfitzmann, B.: Breaking an efficient anonymous channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)
13. Schweisgut, J.: Coercion-resistant electronic elections with observer. In: Krimmer, R. (ed.) EVOTE 2006, 2nd International Workshop on Electronic Voting, Bregenz, Austria, pp. 171–177 (2006)
14. Smith, W.D.: New cryptographic voting scheme with best-known theoretical properties. In: FEE 2005, Workshop on Frontiers in Electronic Elections, Milan, Italy (2005)
15. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: FC 2011, 15th International Conference on Financial Cryptography, St. Lucia (2011)
16. Weber, G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: ARES 2007, 2nd International Conference on Availability, Reliability and Security, Vienna, Austria, pp. 908–916 (2007)
17. Weber, S.: Coercion-Resistant Cryptographic Voting: Implementing Free and Secret Electronic Elections. VDM Verlag, Saarbrücken (2008)
18. Wikström, D.: A commitment-consistent proof of a shuffle. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 407–421. Springer, Heidelberg (2009)

Piracy Protection for Streaming Content in Home Networks

Hongxia Jin[1] and Jeffrey Lotspiech[2]

[1] IBM Alamein Research Center
San Jose, CA 95120, USA
jin@us.ibm.com
[2] Lotspiech.com LLC
Henderson, Nevada, USA

Abstract. In this paper we study content protection techniques to defend against piracy for streaming content in home networks where multiple digital devices are connected into a peer-based cluster and seamlessly work together. We are particularly interested in the anonymous rebroadcasting attack where pirates re-distribute the per-content encrypting key or the decrypted plain content. In literature, to defend against an anonymous attack, content is usually built with different variations. For example, content is divided into multiple segments, each segment comes with multiple variations (e.g., watermarks), and each variation is differently encrypted. Each device only has the key to decrypt and play back one variation per segment through the content. The re-distributed keys can be linked back and used to identify the original devices (terms as *traitors*) who were given those keys and involved in the piracy.

This technology works well for prerecorded content scenarios in which a trusted party outside the device pool can deliberately author the content with multiple variations. However it cannot be applied to a peer-based home network when the streaming content is brought into the home network via a peer device who is not a special trust party and who is not allowed to know the secret keys of other peer devices. On the other hand, the trend of the consumer appetite for digital content is increasingly switching from physical media to streaming and internet consumption. In this paper we have designed the first content protection system that allows a recording device inside the home network to bring the streaming content into the home network in a secure way that devices and only devices in the same home network can playback the recording. More importantly, the recorded content without variations can still be used to obtain forensic information, when anonymous piracy attacks occurs, to identify the source devices that participated in the piracy attack. The identified traitorous devices can be revoked for future content access. The technology described in this paper is used to enable the secure sharing of premium quality High Definition content across a consumer's all audio-video devices at its home networks.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 128–141, 2011.

1 Introduction

Home networks become more and more popular where people connect multiple digital devices together at home. For example, people have multiple audio-video recording and play-back devices in different rooms and connect them into a cluster (or a home entertainment network). The goal of a home entertainment network is to enable people to access their content anytime anywhere. For example, one device may do the recording in one room, another device may play back the recording in another room. To facilitate this, industry standard such as High-Definition Audio-Video Network Alliance (HANA) [1] was formed to provide consumers with a simple way to connect and enjoy High Definition (HD) entertainment anywhere. While consumers want to enjoy the convenience brought by a home network, one of the biggest concern is the security of the content. If it is not secure, the content owners will not allow the copy-righted movies to be placed onto people's home network. Similarly, software providers will not allow their copy-righted software to be downloaded into people's home computer networks. On the other hand, streaming becomes increasingly a way to bring content into a home network. It has the obvious convenience that people do not have to leave their home in order to obtain the content. As one can imagine, the content may be encrypted and the digital devices posses secret device keys that enable them to decrypt and play back the content.

1.1 Piracy Threat for Streaming Content in Home Network

An anonymous attack in the home network scenario is as follows. The attackers compromise one or more sets of device secret keys and set up a server with those keys. They also sell a client with the following value proposition: the client will rip any content in your home network cluster for you, allowing you to make unauthorized copies of purchased content for your friends (or more seriously sell them to anybody on the Internet) or to rip rental content so you can permanently add it to your library after only paying the rental fee.

When evidences of this type of illegal sharing (piracy) are recovered, it is highly desirable to design a content protection system that can detect those compromised device secret keys used in the above pirate server and revoke them for future content access. Such a trace-and-revoke system is what this paper is concerned with. In literature, forensic technology used to detect piracy is generally termed as "traitor tracing". The original devices whose keys are compromised and used in the piracy is called "traitors".

1.2 Existing Traitor Tracing Schemes for Physical Media

"Traitor tracing" for prepared content, (eg, physical media) has been extensively studied. In order to trace traitors, different versions of the content need to be prepared before distributing to devices. Content is divided into multiple segments and some segments are chosen to have multiple differently watermarked and encrypted variations. The different encrypting keys allow different devices to

access content through different pathes. Each different path becomes one content version. The recovered pirated variation encrypting keys or the content version can link back to the actual traitorous devices who were assigned those keys or content versions. Various traitor tracing schemes [3] [8] [9] for anonymous attacks have been designed. Advanced Access Content System (AACS) [5] deployed the "*Sequence Keys*" traitor tracing scheme for prerecorded Blu-Ray DVD disc. All these schemes followed the same paradigm above.

A license agency is responsible for assigning tracing keys to devices and also responsible for detecting traitors using the assigned and recovered tracing keys. For AACS type of system for prerecorded content, the content is prepared with multiple variations by a trusted entity outside the device pool before distributing to all devices/users. This entity has knowledge of every tracing key that was assigned to devices in the system, so it is possible to encrypt the content variations in such a way that every device in the system can only decrypt one variation of the content.

However, one cannot directly apply such a scheme for streaming content. There does not exist a special trusted party (license agency) inside the network cluster. Every device is a peer. As a peer device, the recording device, who records and creates multiple variations for the content, is not allowed to know the secret tracing keys owned by other peer devices. Furthermore, there does not exist an online license agency outside the network cluster that the recording device can contact and do authorization when recording occurs. Therefore it cannot prepare the multiple variations of the recording content in the same way as the license agency in the prerecorded content scenarios and still guarantee that every other devices in the same network can playback the recording.

The main contribution of this paper is that we will present the first content protection system that allows one peer device (not a central trusted party) to make a recording of streaming content in a secure way that all other peer devices in the network can play back the recording. Furthermore, the recorded content without pre-streaming-prepared variations can still be used to obtain useful forensic information against the above anonymous key-redistribution attack. The detected traitorous tracing keys can be revoked/disabled from accessing future content. Our scheme achieves the same traceability as the counter part of the prerecorded content. Our scheme can be used in HANA [1] to provide secure sharing of premium high definition content across audio and video devices in a consumer's home network.

In rest of the paper, in Section 2, we will introduce our "recording keys", "recording key table", and "title key blocks" concepts to use in our content protection system. Then in Section 3 we will present the overall architecture of our content protection system for peer-based home networks. In Section 4 we will show how to assign recording keys in our system and then in Section 5, we will present our nested traitor tracing scheme using recording keys. In Section 6, we will discuss revocation of detected guilty devices, analyze tracing efficiency and also present some optimizations to improve feasibility during implementation. We conclude in Section 7 for future work.

2 Preliminaries

In our system, we assign a sequence number 1...n to each piece of content. When every sequence number is used, it repeats itself from beginning again.

1. Tracing Key K_s
   Each device is assigned a set of tracing keys. Each tracing key corresponds to one content sequence number. For example, for a device, tracing key K_{si} corresponds to its tracing key for content sequence #i. The tracing key assignment is from a matrix with each cell being a randomly generated tracing key. The cells in one column represents all the different tracing keys for one content sequence (eg, movie number one) in the sequence of content. Each device gets assigned exactly one tracing key from each column (eg., for each movie). In other words, each device only knows one tracing key for each content sequence number.
2. Title Key K_t
   The title keys are randomly generated for each content and are the actual keys used to encrypt the content. The entire content may be encrypted by one title key; or the content can be divided into multiple segments, and each segment encrypted by a different title key.
3. Recording Key K_r
   Recording keys are used to encrypt the title keys for the content. Recording keys can be static to a cluster of devices in one home network and be same for all content brought into the home network; or more preferably dynamically created for each streaming content brought into the home network.

   If it is dynamic and specific to each streaming content, the recording key table comes into the home network together with the content during streaming, for example, as a header of the streaming content. The recording keys are brought into the home network in the encrypted format in a recording key table.
4. Recording Key Table
   An encrypted recording key table comes with a piece of streaming content. Suppose that streaming content is assigned to be sequence number #i, in each row of the recording key table, it contains a set of recording keys encrypted by the tracing keys for content sequence number #i. The device can use its assigned tracing key for content sequence number #i to decrypt one row of the Recording Key Table and obtain its set of recording keys for this piece of streaming content.

   Figure 1 shows a sample recording key table for streaming movie #i. Each recording key in row #j is actually encrypted by K_{sij}. A device possessing tracing key Ks_{ij} can go to row #j of the recording key table and decrypt to get its set of recording keys, namely Kr_{j1}, Kr_{j2}...

   The name of "recording key" comes from the fact that for streaming content, it is required that one device makes a recording and all other legitimate devices in the same home network cluster should be able to playback the recording. For this reason, the recording keys must have the property that

Ksi1	E(Kr11)Ksi1	E(Kr12)Ksi1	...
Ksi2	E(Kr21)Ksi2	E(Kr22)Ksi2	...
.			
.			
.			
Ksij	E(Krj1)Ksij	E(Krj2)Ksij	...

Fig. 1. Encrypted Recording Key Table for movie #i

any two devices must share at least one recording key in common. We will discuss more details on recording key assignment in Section 4.

5. Title Key Blocks TKB
   TKB is created by the recording device in the home network when making the recording. For each title key K_t used to encrypt the streaming content, the recording device encrypts that title key with every recording keys it obtains from the recording key table for this content. Each such encrypted title key is an entry of the title key block.

   Figure 2 shows a sample TKB for one title key K_t. It is possible to use multiple title keys to encrypt the content. As one can imagine, different parts of the content may be encrypted using different title keys. For example, a video recording could change the title key every minute of video. In that case, each title key will be encrypted by all the recording keys assigned to the recording device for this content, resulting in multiple TKBs, each TKB looks like the one shown in Figure 2. The recording device can insert the title key blocks as a header for the recording content.

Kr1	E(Kt)Kr1
Kr3	E(Kt)Kr3
.	
.	
.	
Krj	E(Kt)Krj

Fig. 2. Title Key Block for a title key K_t used in a recorded movie

3 Content Protection System for Streaming Content

Figure 3 illustrates an overall architecture of a content protection system for streaming content.

In this overall architecture, there are four types of components in the system, the first one is the license agency, similar to that in the prerecorded content case; and the second one is the streaming content provider. The two other components are inside the peer-based home network corresponding to the two types of devices in the home network, one type of devices can record/download the streaming content (for example, a video recorder and a computer); the other

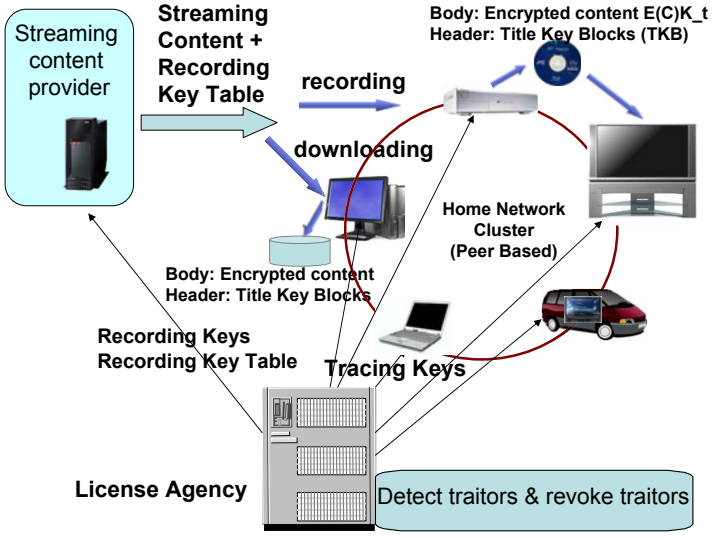


Fig. 3. Architecture for Content Protection system for Streaming Content

type of devices can only playback the recorded content (for example, a high-definition DVD player). Also note that the home network is peer-based. The content streamed by the content provider does not come with multiple variations, instead it is brought into the home network by a recording device inside the home network. The requirement of a home network is that a recording device can record the streaming content in a secure way that only devices inside the same home network can playback the recorded content. Now we will show how components in this system interplay with each other in this architecture.

3.1 License Agency: Manage Secure Content Sharing in Home Network

License agency is a trusted party, it assigns the tracing keys to all devices in a home network cluster. The assignment can be done from a matrix as discussed in last section.

The license agency is responsible for assigning the recording key and creating the recording key table for the content provider. The process takes in the following steps:

1. Assign/create recording keys based on the approach detailed in Section 4.
2. Suppose the current streaming content is sequence number #i, use tracing keys associated with sequence number #i to encrypt the recording keys and create the encrypted recording key table as shown in Figure 1.

We discussed above, in a home network, it requires that one device does the recoding and all other devices in the system can play back the recording. In turn, it requires that recording keys for any piece of streaming content must be created/assigned in such a way that any two sets of the recording keys must share at least one recording key in common. We will discuss the assignment in more details in Section 4.

When piracy occurs, the license agency is responsible for tracing traitors and revoking the guilty tracing keys for future content use. We will discuss these functionalities in Section 5 and Section 6.

3.2 Content Provider: Distribute Streaming Content

As one can imagine, the content provider will stream the content to its users. In addition to the content, it will also provide to the user a recording key table assigned from the license agency for this piece of streaming content. Recall that the recording key table is encrypted by the tracing keys as described in the previous section.

3.3 Recording Devices: Encrypt Streaming Content

A recording device in the home network brings the streaming content into the network cluster. It can record/download the streaming content and bind the content into a secure format. For example, a video recorder may record a streaming movie off the air onto a disc; a computer may download a copy-righted software onto a secure file.

Note that the license agency only exist outside the device cluster and only exist offline. For the operations inside the cluster, there does not exist such a trusted party. When recording operation occurs, the device cannot contact license agency to do authorization. In fact, during the recording and binding, the recording device performs the following steps:

1. Use its tracing key to decrypt the recording key table for this streaming content and obtain its set of recording keys $Kr1, Kr2...Krx$.
2. Randomly pick title keys K_t and use them to encrypt the streaming content.
3. Encrypts each title key with all the recording keys it has K_{ri} for this piece of content and create the Title Key Blocks.

As a result, the newly recorded content is encrypted with the title keys and the TKBs become the header of recorded content. In summary, the recorded content package contains the following:

1. The encrypted recording key table coming together with this streaming content
2. The encrypted (i.e., re-bound) content: $E_{K_t}(content)$
3. The encrypted title key block: $\langle E_{k_{r1}}(K_t), E_{k_{r2}}(K_t), \ldots, E_{k_{rx}}(K_t)\rangle$

3.4 Play-Back Devices: Decrypt Content

When a playback device in the home network cluster tries to playback the recorded content, the decryption process consists of the following steps:

1. Use its tracing key to decrypt the recording key table and obtain its set of recording keys for this piece of content
2. Use one of its recording key to decrypt the Title Key Blocks and obtain the title key K_t
3. Use K_t to decrypt the content and play back the content

Because of the recording key assignment properties which guarantees that a device shares at least one recording key with the recording keys known/used by the recording device during the recoding, the playback device can decrypt at least one of the entries in the TKB for each title key and obtain the title key to playback the content.

4 Recording Keys Assignment

The license agency is responsible for creating/assigning recording keys to a piece of streaming content.

There are various techniques to ensure the above mathematical property for recording key assignment, i.e., two sets of recording keys share at least one key. For example, a randomly-generated code can be filtered to have the desired property. Furthermore, each row can be thought of as a "codeword" and each recording key is thought of as a "symbol". A systematic assignment like a Reed-Solomon code is guaranteed to have the minimum difference. For a specific example, if we need to create 1024 codewords of length 15 (15 recording keys per device) and with 16 different symbols in each column (16 different recording keys in each column), we can use a Reed-Solomon code to create 4096 codewords and any two codewords will differ at at least 13 positions. We can filter those 4096 codewords and get the 1024 codewords among which any two codewords are either one or two symbols (recording keys) in common.

A sample recording key assignment from a matrix is shown in Figure 4. The number of columns corresponds to how many recording keys each device will be assigned. Each row corresponds to how many different recording keys in each column.

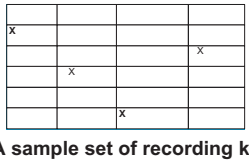


Fig. 4. Recording Key assignment for a movie content

It is worth note that while the above mathematical property has to hold in the normal operational case, it does not have to hold in the forensic case. For that reason, in the forensic case it is possible to assign recording keys using codewords that are maximally apart so as to increase the efficiency of the traitor detection. For example, a Maximal-Distance-Separate code like Reed Solomon code using the assignment shown in Figure 4 can serve the forensic situation very well.

It is also possible to design a recording key assignment that is good for both operational and forensic case. For example, one can have K keys spread over C columns. If $C > K/2$, then the code is guaranteed to be overlapping. However, one can also use a larger K if one can do some filtering. An example with 15 keys spreading over 5 columns can generate 317 unique codewords. The most any single key used was 269 times. If we do not need that many codewords, we can filter further. For example, it is possible to obtain 45 unique codewords with the most-used key used only 20 times. This provides a codeword assignment that satisfies the above mathematical property and yet still has very nice spread of the keys. If the license agency uses this kind of recording key assignment for operational and forensic case, the pirate server will have no way to distinguish between the operational and the forensic situation.

5 Traitor Tracing Using Recording Keys

The license agency is responsible for identification of traitors (guilty devices that use their tracing keys in piracy) and revoke those guilty tracing keys for future use.

As mentioned in Section 1.1, we are mainly concerned with the following style of attack: the attackers compromise one or more sets of device secret keys and set up a server with those keys. They also sell a client that will rip any protected recording you have made: allowing you to make unauthorized copies of protected recording and sell them. The licensing agency wants to detect which devices and their sets of tracing keys are compromised by the pirate server, and revoke those compromised devices tracing keys for future content access.

Our proposed forensic scheme for streaming media is a nested traitor tracing scheme that consists of two sub-schemes, namely Inner Tracing scheme and Outer Tracing scheme. As shown below, the **Inner Tracing** sub-scheme contains *Inner assignment* and *Inner coalition-detection* tasks. The **Outer Tracing** sub-scheme contains *Outer assignment* and *Outer coalition-detection* tasks.

1. *Outer assignment*: How to assign the recording keys and tracing keys to devices
2. *Inner assignment*: how to assign different title key blocks
3. *Inner coalition-detection*: Repeatedly discover pirated title keys, and trace back to the guilty tracing keys
4. *Outer coalition-detection*: Based on the detected guilty tracing keys from Inner Tracing, trace back to the guilty devices that were assigned those tracing keys

In a forensic situation, the licensing agency buys such a client from the pirate server and carefully crafts some forensic content to probe the pirate client and detect what devices have been compromised and used inside the pirate client.

5.1 Inner Tracing Scheme

The goal of Inner tracing scheme is for a particular content sequence number, detect the guilty tracing keys. Recall the tracing keys are assigned from a matrix and each column corresponds to a particular sequence number. Inner tracing scheme needs to detect the compromised tracing key for a particular column.

This Inner Tracing scheme to detect guilty tracing keys is termed as "*forensic analysis based on recording keys*" and is shown in Figure 5. The inner tracing process consists of the following steps.

1. License agency constructs a recording key table for a particular content sequence number #i and encrypts it with those tracing key K_s from column #i of the matrix.
2. (a) License agency builds forensic (recorded) content: pick different title keys K_t to encrypt different content (or different portions of the same content); randomly pick a set of recording key from the above recording key table (for example, from row $\#j$) to encrypt each of the title keys and create the Title key blocks
   (b) Feed the above crafted recorded content to the pirate client for it to decrypt
   (c) The pirate client successfully decrypts the Title Key Block using a recording key it knows and plays back the content
   (d) The content version it plays back reveals which recording key the pirate client has on row $\#j$ to the license agency.
   (e) Iterate the probe by creating another forensic content using a set of recording keys chosen from another row of the above recording key table.

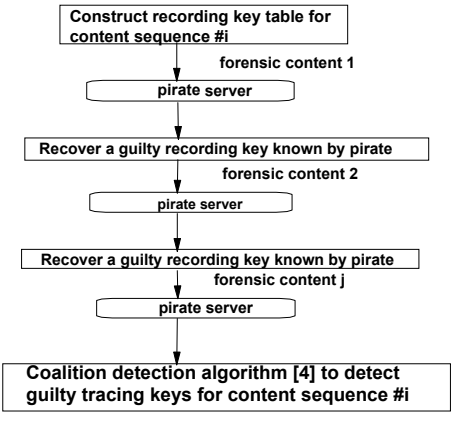


Fig. 5. Inner Tracing based on recording keys to detect guilty tracing keys

3. After recovering enough number of pirated recording keys, use Traitor Detection algorithm [4] to identify the guilty tracing keys for that column. We will briefly explain the algorithm [4] in Section 5.3.

5.2 Outer Tracing Scheme

Our scheme has an **Outer Tracing** sub-scheme called *"Forensic analysis based on tracing keys"* as illustrated in Figure 6. This procedure repeatedly invoke the **Inner Tracing** sub-scheme to detect guilty tracing keys in different content sequence numbers (i.e., columns from the tracing key assignment matrix). When enough number of guilty tracing keys are recovered from different columns, the license agency can again use the coalition detection approach shown in [4] to detect the guilty devices.

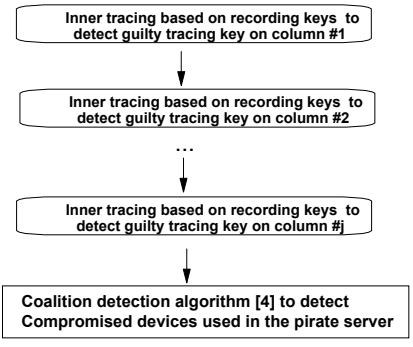


Fig. 6. Outer Tracing based on tracing keys to detect guilty devices

5.3 Coalition Detection Algorithm [4]

In this section, we will briefly explain the coalition detection algorithm that we use in our Inner tracing and Outer tracing scheme. It appeared in [4].

The traitor detection task is the second step of a typical traitor tracing scheme. The first step is how to assign the different keys (or content versions) to different devices. For example, it can be assigned from a matrix, each device gets assigned exactly one key from each column. For example, if a device is assigned a key K_{ij}, it means, for content sequence number #j, this device is assigned key #i for that sequence number. In other words, for each content sequence number, there are multiple (versions of) keys, each device gets exactly one key/version.

When different keys (or content versions) are recovered from piracy on different sequence numbers (i.e., columns), the traitor detection task is to link the recovered keys/versions back to those devices who were originally assigned those keys/versions. The traitor detection is made difficult because of the possibility of attacker collusion. For example, for content sequence number #1, the attackers can redistribute using the version available to colluder #i; when pirating for content sequence #2, the attackers can use and redistribute the version available to colluder #j.

In a typical traitor detection approach, the recovered keys/versions are matched against the keys/versions assigned to the devices. The detection approach will incriminate the devices with the highest number of matchings (sometimes called scores).

In the coalition detection algorithm [4], instead of scoring each individual device, authors proposed to score a set of devices. Their algorithm is based on one important observation: an individual device may score high due to chance alone, while it is much less likely that a coalition (set) of high scoring devices would match all recovered pirate keys/versions due to chance alone.

So the main aim in their algorithm is to find the minimum sized coalition that can explain/match all recovered keys/versions. It is basically the classic Set-Cover algorithm. Given an attack, the tracing algorithm runs by first searching for a coalition of size 1, i.e., a single device. If no such device is found, then the algorithm searches for coalitions of size 2 that explain the attack. If none are found, it searches for coalitions of size 3 that explain it. Eventually, a coalition will be found as long as the number of recovered keys/versions are enough. The authors claimed several advantages of using their algorithm. One of the main advantage is the efficiency measured by the number of recovered keys/versions needed in order to determine the guilty devices with high confidence. Their algorithm achieves almost linear traceability, meaning, for T attackers involved in a collusion attack, their algorithm takes $O(T)$ number of tests/probes (recovered keys/versions) to identify traitors in that coalition. Due to its superior traceability we employ this algorithm in our traitor tracing scheme.

6 Discussion

6.1 Traceability Analysis

As with all other traitor tracing schemes, the traceability of our traitor tracing scheme is measured by the number of forensic testings it takes to probe the pirates and recover forensic feedbacks. In both sub-schemes, we employ the algorithm shown in [4] for the coalition detection task. It takes $O(t)$ number of times to recover forensic feedbacks from pirates when there are t pirates collude in the attack. As a result, the *"forensic analysis based on tracing keys"* needs to be invoked $O(t)$ times, each invocation in turn needs to invoke *"forensic analysis based on recording keys"* $O(t)$ times. Therefore, the total number of probes it takes to detect traitors in a coalition of size t in anonymous attack is $O(t^2)$.

6.2 Revocation after Traitor Detection

As mentioned earlier, the license agency is responsible for assigning the tracing keys to devices; assigning recording keys and creating recording key tables to give to content providers; tracing traitorous devices involved in piracy; and revoking traitorous devices for future content access.

In our system, assigning tracing keys to devices in the cluster is static. The recording keys and recording key tables are dynamic and specific to each content.

In order to disable/revoke those identified traitorous devices and their compromised tracing keys for future content access, the license agency will generate garbage entries associated with those tracing keys when it creates a recording key table for future content. When content provider streams content together with such a recording key table, every guilty device using their tracing keys can only decrypt some garbage (invalid recording keys) out of the recording key tables. Those invalid recording keys will disable the guilty devices from accessing the new content coming together with those recording key tables. That is how revocation works in our system.

6.3 Implementation Considerations

From implementation point of view, there are some optimizations we can do to improve the feasibility of our scheme.

We believe it is possible to improve processing efficiency. For example, one can label or name each recording key entry in the recording key table. These labels are encrypted in the table just like the keys. Then, when a device records a header by encrypting the title key with each of its recording keys, it also stores the labels of those keys in the clear. Then, another device can process the header more rapidly, by searching for labels for which it knows the recording key. Otherwise, the device would have to perform all the decryptions and determine which decryption yields the valid title key by some other means.

We also believe it is possible to reduce the size of the recording key table without reducing security. For example, rather than using full-length keys in the recording table, it is possible to store smaller-length values, and then cryptographically combine those values with the media key to produce a full-length key. For example, if the stored recording keys were 64-bits, and the media key was 128-bits, and the system was using 128-bit AES encryption, the device could expand the 64-bits by concatenating a 64-bit constant defined by the system, and then use the media key to decrypt the resulting value. This "decryption" would be suitably random, and could serve as the actual 128-bit recording key. Thus, the size of the recording key table can be reduced.

7 Conclusion

In this paper we are concerned with piracy protection for the anonymous attack for streaming content in home networks. Unlike traitor tracing schemes for physical media where there exists a trusted party knowing all device secret keys in the system, in a peer-based home network with streaming content, the recording device does not know other peer devices' secret keys, and therefore cannot create the content variations in the recording so that other peer devices can playback the recording later.

In this paper we presented the first content protection system that does not require building multiple variations before the content is streamed. We introduced the concept of "recording keys" that allows one recording device in the

network to make a recording in a secure way that all other devices in the network can playback. We proposed a traitor tracing scheme that makes use of recording keys to gain useful forensic information. Our nested tracing scheme takes $O(t^2)$ probes to the pirate server in order to detect guilty pirates of coalition size t.

As future work, we would like to continue improving the feasibility of the scheme. While this itself is not a concern at the cryptographic traitor tracing scheme level, we would like to improve the practicality of our scheme by combining other approaches including signal processing level technologies.

References

1. http://www.hanaalliance.org/
2. http://www.4centity.com/
3. Jin, H., Lotspiech, J., Nusser, S.: Traitor tracing for prerecordable and recordable media. In: ACM DRM Workshop, Washington, D.C (2004)
4. Jin, H., Lotspich, J., Meggido, N.: Efficient Coalition Detection in Traitor Tracing. In: Proceeding of IFIP International Conference on Information Security 2008, Milan, Italy, September 8-10 (2008)
5. http://www.aacsla.com/specifications
6. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
7. Naor, D., Naor, M., Lotspiech, J.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
8. Fiat, A., Tassa, T.: Dynamic traitor tracing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 354. Springer, Heidelberg (1999)
9. Safani-Naini, R., Wang, Y.: Sequential Traitor tracing. IEEE Transactions on Information Theory 49 (2003)

JITDefender: A Defense against JIT Spraying Attacks

Ping Chen, Yi Fang, Bing Mao, and Li Xie

State Key Laboratory for Novel Software Technology, Nanjing University
Department of Computer Science and Technology, Nanjing University, Nanjing 210093
{chenping,fangyi,maobing,xieli}@nju.edu.cn

Abstract. JIT spraying is a new code-reuse technique to attack virtual machines based on JIT (Just-in-time) compilation. It has proven to be capable of circumventing the defenses such as data execution prevention (DEP) and address space layout randomization(ASLR), which are effective for preventing the traditional code injection attacks. In this paper, we describe JITDefender, an enhancement of standard JIT-based VMs, which can prevent the attacker from executing arbitrary JIT compiled code on the VM. Thereby JITDefender can block JIT spraying attacks. We prove the effectiveness of JITDefender by demonstrating that it can successfully prevent existing JIT spraying exploits. JITDefender reports no false positives when run over benign actionscript/javascript programs. In addition, we show that the performance overhead of JITDefender is low.

1 Introduction

In recent years, attackers have resorted to code-reuse techniques instead of injecting their own malicious code. Typical techniques are Return-oriented Programming (ROP) [21], BCR [12] and Inspector [11]. Different code-reuse attacks launch the attack based on different codebases, including the application, shared libraries or even the kernel. However, all the techniques need to find useful instruction sequence in the codebase, and the task is tedious and costly in practice.

Recently, a new code-reuse attack named JIT (Just-In-Time) spraying was proposed by Blazakis [10]. It reuses JIT-compiled code on the Flash VM's heap in accordance with the attacker's wish to construct the attack. Later, Sintsov published several real-world JIT spraying attacks on Flash VM [26] and further proposed advanced shellcode which leverages the code on Safari's Javascript engine [25]. JIT spraying can circumvent the techniques such as data execution prevention (DEP) [6] and address space layout randomization (ASLR) [6, 9], which are effective for preventing the traditional code injection attacks. The JIT compiler improves the runtime performance of the VM by translating bytecodes into native machine code. A JIT spraying attack is the process that it coerces the JIT compiler to generate native code with the malicious code on the heap of VM, then it exploits the bugs in the browser or its plug-ins to hijack the control and uses the injected malicious code to achieve the attack. JIT spraying attack have become a big threat to Web Security, because the browser often enables dynamic languages (e.g., actionscript and javascript). JIT spraying attacks have two advantages compared with other code-reuse techniques: First, the malicious code is generated by the JIT compiler, so the attacker needs not to piece up useful code snippets for constructing the attack in the codebase. Second, malicious code is generated on the heap.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 142–153, 2011.

This can be combined with heap spraying techniques [29] to increase the probability of the attack.

In this paper, we describe JITDefender, a defense of JIT spraying attacks, which enforces the JIT-code execution control policy for the VM. To defend against JIT spraying attacks, JITDefender changes the VM in the following way: the VM's heap is generally set non-executable so that $W \oplus X$ protection applies. If a specific part of JIT-code has to be executed on the VM based on the definition of the program, the heap is set to be executable, before the code is executed, and immediately set to non-executable afterwards. This paper makes the following contributions:

- We propose JITDefender, an effective technique for defending JIT spraying attacks by controlling the execution of the JIT-code. This techniques distinguishes the benign usage of JIT-code from malicious usage of the attacker.
- We implement and evaluate JITDefender on several commonly used VMs that use JIT compilation (Tamarin flash VM, the V8 javascript engine and Safari's javascript engine). We show that JITDefender can defend against JIT spraying attacks on VMs based on JIT compilation, although the performance overhead of JITDefender is less than 1%.
- We also show that JIT spraying attacks are not only available on flash/javascript VMs, but also available on other VMs based on a JIT compiler, such as QEMU. Our technique is effective for defending against JIT spraying on arbitrary VMs based on JIT compilation.

2 Background: JIT Spraying

JIT Spraying is a code-reuse technique that uses the code generated by the VM based on JIT compilation to launch the attack. By writing the objects using dynamic languages (e.g., javascript, actionscript), the attacker coerces the JIT compiler to generate the malicious code on the heap of VM, and hijacks the control flow to the malicious code snippet. However, different from the existing code reuse techniques, the malicious code does not need to be found within an existing codebase (e.g., libraries, kernel), instead, the attacker predictably defines the objects which are intended to be compiled into the malicious code on the heap of VM, and uses the heap spraying technique to populate the heap with a large number of objects containing the malicious code. Therefore, when the attacker drives the control flow to arbitrary addresses on the heap, with high probability the jump will land inside one of malicious code snippets.

Figure 1 illustrates the JIT spraying attack on Flash VM. First, the attacker defines the actionscript object (`ret` in Figure 1) which contains many uniform statements (`XOR` in Figure 1) with dedicated constructed integers (`0x3c909090` in Figure 1) in the source code. Then the JIT compiler will dynamically translate the source code and generate the native code. In this example, the flash VM will translate the `XOR` as `0x35` and the integer as the original value in the native code. If the attacker carefully constructs the integer value, the native code may be transferred into the malicious code with one byte offset. Suppose the attacker lays out many such objects on the heap, and the attacker can turn the control flow to the malicious snippet (e.g., through a buffer overflow attack), finally this results in the JIT spraying attack.

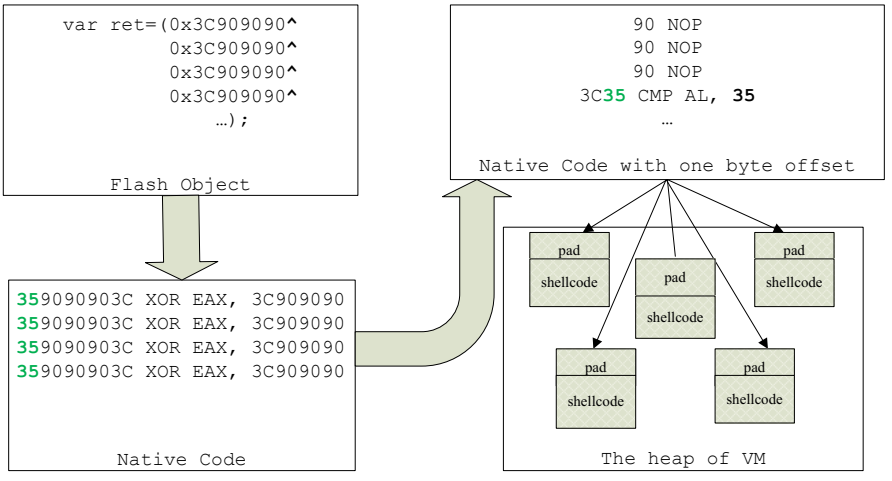


Fig. 1. JIT Spraying Attack

The JIT spraying example illustrated in Figure 1 shows that the code translated by the JIT compiler can be leveraged to launch the JIT Spraying attack. In fact, many other VMs based on a JIT compiler have the same problem. Furthermore, browsers such as IE8 and Chrome are particularly vulnerable to JIT spraying because actionscript/-javascript programs embedded in a web page greatly simplify such attacks.

3 Overview of JITDefender

VMs based on JIT compilation (e.g., Flash VM, Javascript VM) often dynamically translate the source code into the native code on the heap. They necessarily have to mark the page containing the JIT-code as executable and reuse the code repeatedly without re-compiling or interpreting in order to boost the performance. This mechanism implicitly turns the $W \oplus X$ protection off and gives the attacker the opportunity to launch JIT spraying attacks which were illustrated in Section 2. In this paper, we describe the idea of JITDefender, a method to prevent the compiled code from being executed by the attacker. The main idea is to re-enforce $W \oplus X$ protection within the VM. More precisely, we generally mark the native code pages as non-executable. When the VM executes JIT-code, we change the corresponding code pages to executable, after executing the code, the code pages are reset to non-executable again.

When designing JITDefender, we need to identify two points in the the code base of the JIT execution: (1) the *code compilation point*, i.e., the point when the JIT compiler generates the native code, and (2) the *code execution point*, the point when the VM executes the native code. The workflow of JITDefender is that we mark the code pages as non-executable at the first point. Shortly before the second point we mark the pages as executable, and shortly after we mark the pages as non-executable again. This mechanism can be applied to arbitrary VMs based on JIT compiler. Under this protection of

JITDefender, if the attacker hijacks the control flow to the code snippet on the heap for JIT spraying attack the access will be blocked because the VM keeps the code pages non-executable. In fact, JITDefender provides different views of the compiled code for the VM and the attacker with the native code execution control policy.

4 Design and Implementation

In this section, we firstly take the Flash VM as an example to illustrate our method. We identify code parts in the VM that define code compilation and code execution points. Then we demonstrate that our method can be applied to Javascript VM.

4.1 Introduction of the Flash Engine

The source code of flash is written in the actionscript language. Through the different actionscript compilers, the source code is translated into the actionscript byte code (ABC) or the ShockWave File (SWF). The code can be JIT compiled or interpreted on the flash engine. Flash engine handles the ABC or SWF file in the following steps as shown in Figure 2: first it passes through the ABC or SWF files, translates them into the objects which are stored in the object pool. Then it steps into the compiling phase or interpreting phase. In this paper, we only focus on the compiling phase where the JIT compiler fetches the object from the pool, then compiles it into the native code. This is the code compilation point defined above. JIT compiling can be divided into MIR code generation and Machine Code (MD) generation. The native code is stored in the newly allocated heap memory of flash VM. Third, the flash VM provides the loop monitoring whether there is the compiled code on the heap. If so, it will execute the native code. This is the code execution point defined above. In order to prevent JIT spraying attack,

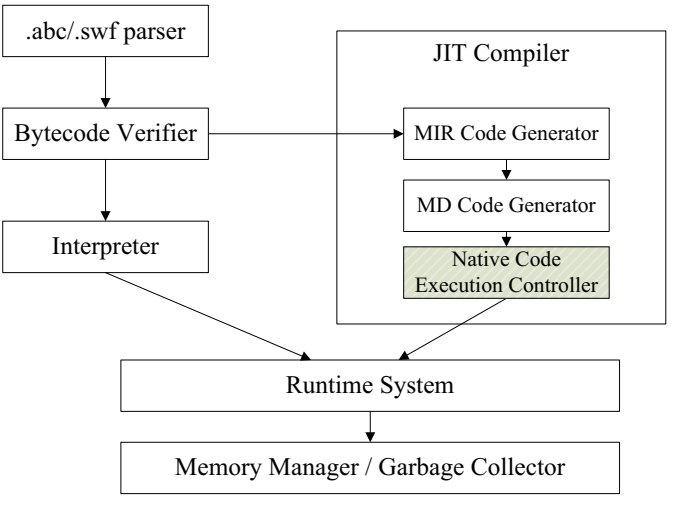


Fig. 2. Tamarin Flash Engine

we manipulate the *Native Code Execution Controller* on VMs, which is between the MD code generation and Runtime code execution. In the following subsection, we will illustrate the mechanisms implemented in Flash/Javascript VMs in details.

4.2 Adapting the Flash Engine

In Flash Engine Tamarin, when an ABC or SWF file is loaded, the JIT compiler will translate the source code into native code at the unit of a function. More specifically, Tamarin uses the class `MethodInfo` to store the information of the functions that can be executed by the VM, including user-defined functions, native functions and so on. Another key class named `CodeMgr` is used to manage memory for compiled code, including the code itself (in a `nanojit::CodeAlloc`), and any data with code lifetime (in a `nanojit::Allocator`), such as debugging info and inline caches. In order to set the attributes of the code pages, we should get the the compiled code information at the unit of function. Therefore, we add the variable `CodeMgr* mgr` in the `MethodInfo` class. It will provide us the convenience for setting attributes of the code pages, because we can get the information of the function including the native code generated by this function. We now give more details how to set the page attributes.

First, we need to get related information at the code compilation point when native code has been generated. We find that, after Tamarin generates the compiled code for one function, it will store the compiled code information in the `codeMgr` which is the member variable of the class `PoolObject`, and `PoolObject` is a container for the pool of resources decoded from an ABC file. Therefore, we will transfer the compiled code information into the new variable `mgr` we defined in `MethodInfo`. Then we extract the start and end address of the compiled code memory, and invoke the function named `VMPI_setPageProtection` to set the related pages as non-executable.

Second, we need to find the code execution point. As mentioned in Section 3, we should set the related code pages as executable before executing the compiled code, and after executing the compiled code, we set the compiled code pages as non-executable again. In the Tamarin flash engine, the function `coerceEnter` is used to execute the compiled code, at the end of this function, the specific handler function is invoked by `endCoerce(argc, ap, ms)`. As such, we leverage the function `VMPI_setPageProtection` to set the related pages as executable, and after executing the function, we use the same function to set the related pages as non-executable. Note that we calculate the related pages based on the recorded information of `mgr` in the `MethodInfo`.

The method mentioned above can successfully protect the compiled code on the heap of the VM. Because the compiled code can be only executed by the VM itself, but can not be executed by the attacker.

4.3 Javascript Engine

Sintsov [25] shows that JIT Spraying can be also mounted on the Javascript Engine in Safari [4]. Similar to the flash engine, the attacker can construct the Javascript object using the `XOR` operation with the specific integer operands. Then the Javascript Engine will compile the Javascript objects into the native code which contains the malicious

code. We find that the same problem occurs in the V8 Javascript engine – the javascript engine of Chrome [18]. We could leverage a known buffer overflow [1] to launch a JIT spraying attack. In this section, we demonstrate that JITDefender can be implemented on both Javascript engines.

V8 Javascript Engine. As we mentioned in Section 3, we need to identify the code compilation and the code execution point. The V8 Javascript engine provides two API function for compilation and execution respectively. The `Compile` function is used to compile the Javascript program into the native code on the heap, and the `Run` function is used to execute the compiled native code.

In the compilation phase, the V8 Javascript engine parses the Javascript files and divides the code into two parts, one is the specific function such as the `eval`, the global function, and other functions are regarded as the shared function. Then it compiles the Javascript code into the native code according to the function categories. And the native code is stored as `SharedFunctionInfo` at the unit of the function. `SharedFunctionInfo` is the child class of `HeapObject`, which maintains the information of the Javascript objects.

In the execution phase, V8 gets the compiled code by using the `Code::GetCode` method, and finds the entry to the code by the `Code::entry` method, then it jumps to the code entry to execute the code on the heap using `Execution::Call`. Note that the class `Code` is the child class of `HeapObject`, it contains the native code generated at the compilation point. Similar with the flash engine, the compiled code is laid out on the heap and its page is marked as executable, which is independent of whether the code is executed by the Javascript engine or not.

We applied our code control policy to the V8 javascript engine. First, at the end of `Compiler::Compile`, we use the Windows API function `VirtualProtect` to set the compiled code pages as non-executable. Then before the `Execution::Call`, we use `VirtualProtect` to set the code pages as executable, and after executing the compiled code, we set the code pages as non-executable again.

Safari's Javascript Engine. Similar to the V8 Javascript Engine, at the code compilation point, Safari's Javascript Engine compiles the Javascript code into the native code according to the function definition. The native code is saved in the structure of `JITcode`. At the code execution point, Safari Javascript Engine will query the native code base, invoke the entry to the current executed function and then execute it. We modify Safari's Javascript Engine (JavascriptCore) at these two points: First, Safari's Javascript Engine gets the JITed code by the method `JIT::compile` in memory space in the form of `JITcode`. Then we set the code pages as non-executable. Second, the Javascript Engine executes the JITed code by using the method `JITStubCall::call`. Before we invoke this function to execute the JIT-code, we first set the code pages as executable. After we executed the code, we reset the code page as non-executable again.

5 Evaluation

In this section, we describe the experimental evaluation of our JITDefender prototype. First, we test JITDefender's ability to dynamically defend the JIT spraying attack.

Second, we measure the performance overhead of JITDefender. The evaluation is performed on an Intel Pentium Dual E2180 2.00GHz machine with 2GB memory and Windows 7. Tested programs are compiled by Microsoft Visual Studio 2008.

5.1 Effectiveness

Since JIT spraying is a new attack, there are little attack samples published. Therefore we used existing JIT spraying attacks published by Sintsov for Tamarin flash engine and Safari's Javascript engine and wrote samples for the V8 javascript engine by ourselves. More specifically, we chose two JIT spraying attack samples written by Sintsov to evaluate Tamarin flash engine: "SAP-Logon7-System" [24] and "QuikSoft-STAGE0" [23]. The two JIT spraying attacks leverage the buffer overflow vulnerability in SAPGUI 7.10 ActiveX and Oracle Document Capture (EasyMail Objects EMSMTP.DLL 6.0.1) ActiveX Control respectively, and can successfully launch attacks on IE8. Note the two original JIT spraying attacks leverage Flash Player 10's flash engine [26]. In our experiment, we use the Tamarin Flash engine instead. For Safari, we chose the JIT Spraying attack "Safari_parent_close_sintsov" published by Sintsov [22]. It exploits the vulnerabilities in Safari 4.0.5 `parent.close()` to launch attacks. In addition, in order to test the effectiveness of our tool on the V8 Javascript engine, we wrote one JIT spraying code by leveraging the buffer overflow "SaveAs" in Chrome [1]. We named the attack as "SaveAs-JITSpray". With all the attacks mentioned above, we tested the effectiveness of JITDefender for detecting the JIT spraying attacks, including Tamarin, V8, and Safari's javascript engine. Experimental results in Table 1 show that JITDefender can successfully defend JIT spraying in VMs based on JIT compilation. We also tested JITDefender on benign code, namely the performance benchmarks embedded in the JIT VMs. The actionscript/javascript programs are listed in Table 2. Overall, we found no false positives.

Table 1. JIT Spraying Attacks Tested on JITDefender

VMs	JIT Spraying Attacks	LOC(K)	Description	JITDefender
Tamarin	SAP-Logon7-System [24]	3K	SaveViewToSessionFi ActiveX Buffer Overflow	✓
	QuikSoft-STAGE0 [23]	3K	SubmitToExpress ActiveX Buffer Overflow	✓
V8	SaveAs-JITSpray	3K	"SaveAs" Buffer Overflow in Chrome	✓
Safari's JS Engine	Safari_parent_close_sintsov [22]	3K	Safari 4.0.5 parent.close() (memory corruption)	✓

5.2 Performance Overhead

We also measured the performance overhead of JITDefender. Because JITDefender modifies the JIT-code page attributes of the VM at runtime, it will bring some overhead to the VMs. In this section, we chose the three VMs based on JIT compilation to evaluate the performance overhead of JITDefender, including Tamarin Flash Engine, Safari's Javascript Engine and V8 Javascript Engine. Table 2 shows the performance overhead of JITDefender when it is applied to the VMs when executing the actionscript/javascript programs. For each VM, we run the benchmark of it, and compare the time costs when the tested program running on the original VMs and the modified VMs. By comparison, we can see that JITDefender has less than 1% costs on VMs. Generally speaking, the performance overhead is proportional to the number of function chunks in the program.

Table 2. Performance Overhead of the JIT VMs under JITDefender

VMs	Benchmarks	LOC(K)	Original VM	JITDefender	Performance Overhead
Tamarin	SOR.as	3	72.844s	72.999s	0.2%
	Heapsort.as	3	9.980s	10.325s	3.5%
	SparseMatmult.as	4	0.958s	0.983s	2.6%
	FFT.as	5	10.334s	10.552s	2.1%
	Series.as	6	4.661s	4.763s	2.2%
	LUFact.as	12	27.827s	27.989s	0.6%
	Moldyn.as	14	6.101s	6.276s	2.9%
	Crypt.as	15	0.314s	0.322s	2.5%
	RayTracer.as	22	1.566s	1.573s	0.4%
	Euler.as	81	0.246s	0.249s	1.2%
	Average		13.483s	13.603s	0.9%
V8	crypto.js	48	5.189s	5.196s	0.1%
	richards.js	16	2.107s	2.112s	0.2%
	deltablue.js	26	2.100s	2.113s	0.6%
	raytrace.js	28	2.129s	2.134s	0.2%
	earley-boyer.js	195	5.198s	5.213s	0.3%
	regexp.js	105	4.274s	4.287s	0.3%
	splay.js	11	3.575s	3.629s	1.5%
	Average		3.510s	3.526s	0.5%
Safari's JS Engine	crypto.js	48	5.336s	5.419s	1.6%
	richards.js	16	2.277s	2.288	0.5%
	deltablue.js	26	2.246s	2.301s	2.4%
	raytrace.js	28	4.309s	4.339s	0.7%
	earley-boyer.js	195	12.496s	12.792s	2.4%
	regexp.js	105	15.944s	15.959s	0.1%
	splay.js	11	4.911s	4.918s	0.1%
	Average		6.788s	6.859s	0.1%

This is because the more function chunks of the program exist, the more frequently JIT VMs will transfer the JIT objects into different functions' native code and execute these codes individually. Since we modify the code pages' attributes, the more function chunks of the programs there are, the more performance overhead will be introduced to program's execution. Take the experimental results in Table 2 for instance, the test case "Heapsort.as" uses recursion techniques to sort arrays. It frequently invokes the small function `NumHeapSort`, which performs a heap sort on an array. Therefore, when the Tamarin flash engine compiles "Heapsort.as" into native code, the code pages will be marked as "non-executable" or "executable" alternatively. This is the reason why the performance overhead of executing "Heapsort.as" under JITDefender is 3.5%, which is more than the average.

6 Discussion

6.1 JITDefender on Other VMs

QEMU [3] is the commonly used CPU-emulator, which leverages the JIT techniques. We discovered that the programs running in QEMU will cast their code on the heap

and mark it as executable. We note that JIT spraying attacks may be constructed on it too. To perform such an attack, we need to install a VM on QEMU. Within the VM, we need to construct a malicious program, which contains the shellcode. Then we fork several processes for running the program that keep on spraying malicious code on the heap of QEMU. In order to construct the attack, we need to leverage one of the bugs in QEMU and drive control to the malicious code [28]. Since QEMU is used as CPU-emulator in KVM [2], this effectively means that the approach can be leveraged to construct an attack on Cloud Computing. So JIT spraying may threaten the security of the Clouds. Similarly with to flash and javascript engine, it should be possible to implement JITDefender on QEMU. In fact, we believe JITDefender can be applied to arbitrary VMs based on a JIT compiler to defend against JIT Spraying exploits.

6.2 Circumventing JITDefender

The method proposed in this paper addresses the problem that most Just-In-Time compilers leave a large window of opportunity for exploiting code "sprayed" into the compiled code. We reduce the window to the minimum time so that only the VM based on JIT compilation can set the compiled code as executable at page granularity. People may argue that is it possible that the attacker exploits the vulnerability within the window which spans from VM setting the page as executable to VM executing the code on the page. We think this potential JIT spraying attack is feasible in theory but we did not find it in practice. The potential JIT spraying attack is that the malicious code is on the same page with the code that is executing on VM when the attacker exploits. In that case, JITDefender will keep the page as executable, as such, JIT spraying will make JITDefender ineffective. Although we did not find such attacks till now, we consider the attack is possible. For example, the attacker introduces the delay method (e.g., loop) in certain function to keep the code executing and the page containing the malicious code. Even worse, if the program is asynchronous, for example, it uses multi-threaded methods to load multiple objects on the heap, and keep all the pages executable by the delay method, it will give a big opportunity for the attacker to circumvent JITDefender. Although nobody has proposed the attack till now, in theory, it will be a threat to JITDefender. To counter the attack, we consider to add some optimization to the VM, for example, we can pre-calculate the `XOR` operation, and get the result directly without translating its operands on the heap.

7 Related Work

7.1 Heap Spraying Defenses

Heap spraying [29] is a technique that will increase the probability to land on the desired memory address. The act of spraying simplifies the JIT spraying attack and increases its likelihood of success. There are several defenses specifically designed against heap-spraying attacks [20, 16, 14]. Nozzle [20] is the countermeasure specifically designed against heap-spraying attacks by the analysis of the contents of any object allocated by the Web browser. Similarly, Egele et al. [14] propose an emulation method to detect heap spraying attacks with drive-by downloads. Based on libemu [5], it emulates

the code download and checks whether there is malicious code. Bubble [16] is the Javascript engine level countermeasure against Heap-spraying attacks. It introduces the diversity of the heap by inserting special interrupting values in strings at random positions when the string is stored in memory and removing them when the string is used by the application. All the existing Heap spraying defenses all rely on the assumption that the malicious code is introduced from the outside (network, keyboard), therefore they may be circumvented by JIT spraying attack.

7.2 JIT Spraying Mitigation

Concurrently and independently, Bania [8] proposed a heuristic detection method, based on the assumption that JIT spraying attacks use arithmetic operations. They detect JIT spraying by calculating the number of bad instructions. However, JIT spraying may not use the arithmetic operations to generate the malicious code. Tao et al. [27] propose the code randomization techniques on VMs based on JIT compilation. However, it has 5.9% space and 5% runtime overhead. In addition, they currently implemented the prototype on V8 engine only. Our techniques are quite different compared to theirs. First, we do not assume any particular form of JIT spraying attack (e.g., using `XOR`). Second, our method can prevent JIT spraying with less performance overhead (less than 1%). Third, our method has been implemented and tested on different VMs, and proved to be easily deployed on other VMs in practice. Most recently, de Groef et al. [17] proposed a kernel patch JITsec that defeates JIT spraying based on testing certain restrictions when invoking a system call. Different from JITDefender, JITsec can only defeat the malicious code that uses the system call not general code.

7.3 Other Defenses

Most recently, Payer [19] summarized the different attack types and defenses. Generally speaking, all the attacks leverage software bugs and maliciously craft the control or non-control objects to achieve malicious behavior. To defeat these attacks, researchers proposed CFI [7] and DFI [13] that protect against sensitive objects. XFI [15], a kernel module, leverages static analysis for the guard and checks the jump's target, in addition, it uses two stacks to guarantee the return address. There are intrinsic differences between XFI and JITDefender. First, JITDefender prevents the code snippet in the function from being reused from outside, while XFI controls the function callsites and prevents it being maliciously invoked. Second, JITDefender prevents the JIT spraying attack, while XFI prevents the code injection attacks.

8 Conclusions

In this paper, we present the design, implementation, and evaluation of JITDefender, a tool for defeating JIT spraying attacks. JITDefender applies code execution control on the VMs, and to the best of our knowledge, it is the comprehensive defense for JIT spraying. The evaluation of JITDefender shows that it has no false positives, and the performance overhead is low.

Acknowledgements

We thank our shepherd Felix Freiling for his help on the final paper. We also thank the anonymous reviewers for their constructive and helpful feedbacks and suggestions. This work was supported in part by grants from the Chinese National Natural Science Foundation (60773171, 61073027, 90818022, and 60721002), the Chinese National 863 High-Tech Program (2007AA01Z448), and the Chinese 973 Major State Basic Program(2009CB320705).

References

1. Google chrome 0.2.149.27 'saveas' function buffer overflow vulnerability, http://seclists.org/bugtraq/2008/Sep/70
2. KVM, www.linux-kvm.org/
3. QEMU, http://wiki.qemu.org/Main_Page
4. The Webkit open source project, webkit.org/
5. x86 shellcode detection and emulation, http://libemu.mwcollect.org/
6. The Pax project (2004), http://pax.grsecurity.net/
7. Abadi, M., Budiu, M., Ligatti, J.: Control-flow integrity. In: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS), pp. 340–353. ACM, New York (2005)
8. Bania, P.: JIT spraying and mitigations (2010), http://arxiv.org/abs/1009.1038
9. Bhatkar, E., Duvarney, D.C., Sekar, R.: Address obfuscation: an efficient approach to combat a broad range of memory error exploits. In: Proceedings of the 12th USENIX Security Symposium, pp. 105–120 (2003)
10. Blazakis, D.: Interpreter exploitation. In: Proceedings of tth USENIX Workshop on Offensive Technologies (WOOT 2010), pp. 1–9 (2010)
11. Kolbitsch, C., Holz, T., Kruegel, C., Kirda, E.: Inspector gadget: Automated extraction of proprietary gadgets from malware binaries. In: Proceedings of the 30th IEEE Symposium on Security and Privacy, pp 29–44 (2010)
12. Caballero, J., Johnson, N.M., McCamant, S., Song, D.: Binary code extraction and interface identification for security applications. In: Proceedings of the 17th Annual Network and Distributed System Security Symposium (2010)
13. Castro, M., Costa, M., Harris, T.: Securing software by enforcing data-flow integrity. In: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, vol. 7, p. 11. USENIX Association, Berkeley (2006)
14. Egele, M., Wurzinger, P., Kruegel, C., Kirda, E.: Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In: Flegel, U., Bruschi, D. (eds.) DIMVA 2009. LNCS, vol. 5587, pp. 88–106. Springer, Heidelberg (2009)
15. Erlingsson, U., Valley, S., Abadi, M., Vrable, M., Budiu, M., Necula, G.C.: XFI: Software guards for system address spaces. In: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, vol. 7, p. 6. USENIX Association, Berkeley (2006)
16. Gadaleta, F., Younan, Y., Joosen, W.: BuBBle: A javascript engine level countermeasure against heap-spraying attacks. In: Massacci, F., Wallach, D., Zannone, N. (eds.) ESSoS 2010. LNCS, vol. 5965, pp. 1–17. Springer, Heidelberg (2010)
17. de Groef, W., Nikiforakis, N., Younan, Y., Piessens, F.: Jitsec: Just-in-time security for code injection attacks. In: Benelux Workshop on Information and System Security (WISSEC 2010), pp. 1–15 (2010)

18. Google Inc.: V8 javascript engine, code.google.com/apis/v8/intro.html
19. Payer, M.: I control your code attack vectors through the eyes of software-based fault isolation. In: 27C3 (2010)
20. Ratanaworabhan, P., Livshits, B., Zorn, B.: Nozzle: A defense against heap-spraying code injection attacks. In: Proceedings of 18th USENIX Security Symposium (2009)
21. Shacham, H.: The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86). In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), pp. 552–561. ACM, New York (2007)
22. Sintsov, A.: JIT spraying attack on safari, http://www.exploit-db.com/exploits/12614/
23. Sintsov, A.: Oracle document capture (easymail objects emsmtp.dll 6.0.1) activex control bof - JIT-spray exploit, http://dsecrg.com/files/exploits/QuikSoft-reverse.zip-reverse.zip
24. Sintsov, A.: SAP GUI 7.10 webviewer3d Activex - JIT-spray exploit, http://dsecrg.com/files/exploits/SAP-Logon7-System.zip
25. Sintsov, A.: JIT-spray attacks & advanced shellcode (2010), http://dsecrg.com/files/pub/pdf/HITB%20-%20JIT-Spray%20Attacks%20and%20Advanced%20Shellcode.pdf
26. Sintsov, A.: Writing JIT-spray shellcode for fun and profit. In: Technical Report of Digital Security (2010)
27. Tao, W., Tielei, W., Lei, D., Jing, L.: Secure dynamic code generation against spraying. In: CCS 2010 Poster, pp. 738–740. ACM, New York (2010)
28. Wang, T.: Integer overflow on QEMU, http://lists.nongnu.org/archive/html/qemu-devel/2008-08/msg01052.html
29. Wikipedia: Heap spraying (2010), http://en.wikipedia.org/wiki/Heap_spraying

Retrofitting Security in COTS Software with Binary Rewriting

Pádraig O'Sullivan [1], Kapil Anand [1], Aparna Kotha [1], Matthew Smithson [1], Rajeev Barua [1], and Angelos D. Keromytis [2]

[1] Electrical and Computer Engineering Department, University of Maryland
[2] Department of Computer Science, Columbia University

Abstract. We present a practical tool for inserting security features against low-level software attacks into third-party, proprietary or otherwise binary-only software. We are motivated by the inability of software users to select and use low-overhead protection schemes when source code is unavailable to them, by the lack of information as to what (if any) security mechanisms software producers have used in their toolchains, and the high overhead and inaccuracy of solutions that treat software as a black box.

Our approach is based on *SecondWrite*, an advanced binary rewriter that operates without need for debugging information or other assist. Using SecondWrite, we insert a variety of defenses into program binaries. Although the defenses are generally well known, they have not generally been used together because they are implemented by different (non-integrated) tools. We are also the first to demonstrate the use of such mechanisms in the absence of source code availability. We experimentally evaluate the effectiveness and performance impact of our approach. We show that it stops all variants of low-level software attacks at a very low performance overhead, without impacting original program functionality.

1 Introduction

Despite considerable research and work on programmer education and tools, programming language and compiler support for security, hardware and operating system features, low-level software vulnerabilities remain an important source of compromises and a perennial threat to system security. While other sources of vulnerability have emerged more recently, such as SQL injection, cross-site scripting (XSS) and cross-site request forgery (XSRF), binary-level vulnerabilities continue to be discovered in very popular software and to be exploited for fun and profit [12].

The lack of convergence to a comprehensive solution can be attributed to several factors, consisting of a mix of the technical and non-technical. At the core, there exists a fundamental dichotomy in the capabilities and motivation of producers and consumers of software, vendors and end-users/administrators respectively. On the one hand, software producers are probably in the best position to both proactively and reactively prevent and mitigate such vulnerabilities: they have access to the source code, the compiler tool chain, and the developers themselves. As a result, they can apply security mechanisms that offer high coverage and effectiveness at low overhead, because they are

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 154–172, 2011.

applied at the point where the most semantic knowledge about the program and the code is available. On the other hand, it is software consumers that face the risk and bear the costs of compromise due to software vulnerabilities and are the most motivated to take action, often localized, to mitigate a newly discovered vulnerability. However, consumers often only have access to the program binary and configuration files. Thus, absent vendor patches (which can often take a long time and may contain bugs [32]) consumers can only use security mechanisms that treat the software as a black box. Inevitably, such mechanisms resort to isolation (*e.g.,* through a virtual machine) or to behavioral detection (*e.g.,* system call monitoring), with attendant costs, complexity and risk. Even security-conscious software consumers often cannot properly evaluate the risks they face because they do not know what security mechanisms, if any, a producer has used in their development process and tool-chain [22].

We present a new mechanism based on advanced binary rewriting that seeks to bridge the gap between incentive/motivation and capabilities on the consumer side. Our approach allows end users to retrofit powerful security mechanisms into third-party, binary-only software. These mechanisms are well-known, and some of them have been *partially* integrated in separate tools and development environments (*e.g.,* ProPolice in *gcc* and the optional */GS* flag in Visual Studio). Our system allows end-users to ensure that the software they run on their systems uses any and all such features, regardless of the choices or capabilities of vendors[1]. Furthermore, our approach allows end-users to selectively apply different defense mechanisms to different parts of the program, based on their own analysis, risk assessment, and knowledge of potential or actual vulnerabilities in the code. Essentially, we provide the nearly the same self-defense capabilities that open-source software users *can* utilize to users of binary-only software[2].

The contributions of this paper are twofold. First, we present a powerful binary-rewriting framework in the context of software security. Specifically, we investigate the ability of such a system to retrofit known invasive, powerful and low-overhead security mechanisms to program binaries, in the absence of source code or even debugging symbols. Second, we evaluate the effectiveness and efficiency of our scheme and of the retrofitted security mechanisms, as compared to other ways in which these and similar security mechanisms can be applied to software. We conclude that a system such as ours would enable software consumers to protect themselves at the same level of effectiveness as if vendors had taken similar steps (*i.e.,* used the same security techniques) and at equally low overhead. Thus, we believe that we have removed a significant factor in improving the overall security posture of systems against low-level software compromises.

An additional contribution of this paper is that we have carefully chosen a set of complementary and effective schemes that, taken together, achieve the goal of defending against all types of buffer overflow attacks at the lowest combined run-time cost. The totality of our schemes protect against buffers on the global, stack, and heap segments from overflowing onto a variety of (usually code) pointer locations that are

[1] Not all development tool-chains support a given security feature, while vendors and products are often intimately tied to them. As a result, there is considerable reluctance by vendors to switch to a "better" compiler, for example, even if such existed.

[2] Just because open-source software users *can*, does not mean they generally *do* assess or modify/secure their installations.

vulnerable to attack, including return addresses, function pointers, indirect branch pointers, *longjmp* buffers, and base pointers. This is an important practical contribution in itself, as this is the first solution in the literature to retrofit a comprehensive set of protections against buffer overflow attacks, which are still very common, into arbitrary new and legacy binaries. We intend to make this tool available publicly soon.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work in binary rewriting and binary-only software security mechanisms. Section 3 presents background on binary rewriting, and how rewriting relates to security. We describe the methods we have chosen in Section 5, and discuss experimental results in Section 6. We conclude with our thoughts for future work in Section 7.

2 Related Work

Our work is related to many techniques that attempt to defend against attacks on vulnerabilities in applications. In this section, we elaborate on some of the pieces of work most closely related to ours. We present the various attack techniques utilized by attackers that are relevant for this research and then we go on to present various techniques proposed for mitigating these attack techniques. We also briefly discuss related work in binary rewriting.

2.1 Catalog of Attack Techniques

Buffer Overflow Attacks. A buffer overflow refers to a situation that can occur when code writes into a bounded array, or buffer, and the writes are not correctly guarded against overflow. Data copied into the buffer whose length is larger than the buffer's size is referred to as a buffer overflow. Writes into a buffer that are not correctly guarded may overwrite and corrupt a variety of vulnerable locations that may also be stored nearby the buffer, including return addresses, base pointers, function pointers, and *longjmp* buffers. Although buffer overflows have historically most often occurred on the stack, they are also possible on heap and global segment buffers. For example a global buffer's overflow may overwrite a function pointer or *longjmp* buffer also in the global segment.

Buffer overflow attacks work by changing the value of the code pointer stored in vulnerable locations such as return addresses, function pointers and *longjmp* buffers. The code pointer is overwritten by a new value pointing to code of the attacker's choice. Base pointers are also vulnerable even though they are not code since they can be used for attacks [31]. A return address attack was first described in detail by AlephOne in 1996 [1]. However, attacks of this kind date back to before 1988 when the technique was used in the *fingerd* exploit of the Morris worm.

Commonly, an attacker would choose their input data so that the machine code for an attack payload would be present at the modified return address. When the vulnerable function returns, and execution of the attack payload begins, the attacker has gained control of the behavior of the target software. The attack payload is often called shellcode, since a common goal of an attacker is to launch a command line interpreter (referred to as a shell in UNIX like environments) under their control.

Return-to-libc Attacks. As an alternative to supplying executable code (referred to as direct code injection), an attacker might be able to craft an attack that executes existing

machine code (indirect code injection). This class of attacks has been referred to as jump-to-libc or return-to-libc (arc injection [9] has also been used to refer to this class of attacks) because the attack often involves directing execution towards machine code in the standard C library (libc) [9]. The standard C library is often the target for attacks of this type since it is loaded in nearly every UNIX program and it contains routines of the sort that are useful for an attacker. This technique was first suggested by Solar Designer in 1997 [27]. Attacks of this kind can evade defense mechanisms that protect the stack such as stack canaries and it is also effective against defenses that only allow memory to be writable or executable.

Traditionally, attacks of this kind have targeted the system function in the standard system library which allows the execution of an arbitrary command with arguments. In this case, the return address of a vulnerable function would be modified to point to the address of the system function which would then be executed with attacker supplied arguments. Since the system function executes a command on a system, if an attacker can control the arguments to this function, they could execute an arbitrary command on the system under attack. However, recent attacks have been demonstrated which do not depend on calling functions in the standard C library.

2.2 Catalog of Defense Techniques

Compile Time Defenses. StackGuard [8] places a 'canary' (a memory location) on the stack between local variables and the return address. This canary value is designed to warn of stack corruption since validating the integrity of the canary value is an effective means of ensuring that the function return address has not been corrupted. Microsoft's compiler also supports the insertion of stack canaries with the /GS option.

ProPolice [10] is similar to StackGuard in that it places a canary value on the stack. However, ProPolice also changes the stack layout to place arrays and other function-local buffers above all other function-local variables. Copies of all function arguments are also made into new, function-local variables that also sit below any buffers in the function. As a result, these variables and arguments are not subject to corruption through an overflow of these buffers.

PointGuard [7] protects all code pointers within a program. The defense consists of encrypting pointer values in memory and only decrypting the pointers when they are loaded into CPU registers. The encryption key used is a randomly generated during process creation and is thus unknown to an attacker. Without knowledge of the encryption key, an attacker can not modify any value stored in memory. As a result, pointer values are not subject to corruption.

StackGuard, PointGuard, and ProPolice involve compile-time analysis and transformation. Thus, unless the source code for an application is available, these techniques can not be used thereby hindering the ability to easily deploy these techniques. In practice only the developer can use these defenses, and only if the compiler his or her organization uses supports it. Our techniques do not suffer from this drawback since they can be easily deployed on any binary produced from any source language and compiler, by not only the developer, but the end-user as well.

Instruction Set Randomization. Instruction-set randomization [5] is a technique for protecting against buffer overflows (and many kinds of code injection attacks). This

approach randomizes the underlying system's instructions so that foreign code injected by an attacker would fail to execute correctly since the attacker does not know the instruction set of the target system. However, as mentioned by the authors in [5], the main drawback of this technique as applied to binary code that it needs specialized hardware support in the processor. Thus, even though instruction-set randomization offers a strong defense against buffer overflow attacks the fact that unless it is supported by specialized hardware, it incurs significant overheads means that it is unlikely to see adoption in practice for the foreseeable future.

Strata (a dynamic binary translation framework) and Diablo (a link-time binary rewriter) were used to implement instruction set randomization [16]. Diablo is used to prepare a binary for string encryption and introduce the information necessary to detect foreign code. Strata is then used to provide the necessary virtual execution environment for safe execution. The main contribution of this work is that the instruction-set randomization implementation is efficient while requiring no special hardware support. However, the runtime overheads reported are still high because of the necessary software ISA translation at run-time, and the inherent overheads of a dynamic translator. These run-time overheads and likely to limit the practical adoption of such a system. Moreover any user of a dynamic binary rewriter must install it in addition to the application desired, making it inconvenient to use.

The static (off-line) binary rewriter we use suffers from none of these issues. No special hardware is required to utilize a binary rewriter and overheads are relatively low since no ISA translation is done, and since no dynamic translator is used, no additional software in addition to the application is needed for execution. In our system, if an original binary was compiled without optimizations, we often see a significant run-time improvement when rewritten.

Address Space Layout Randomization. Address Space Layout Randomization (ASLR) can be seen as a relatively coarse-grained form of software diversity. ASLR shuffles, or randomizes, the layout of software in the memory address space. The common implementation of this scheme is at the OS level. Thus, when a process is launched the address space layout of the process will be different from a previous invocation of the same process. It is effective at preventing remote attackers that have no existing means of running code on a target system from crafting attacks that depend on addresses. ASLR is not intended to defend against attackers that are able to control the execution of a piece of software; it is mainly intended to hamper remote attackers from attempting to use the same attack repeatedly. Finally, its utility on 32-bit architectures is limited by the number of bits available for address randomization [25].

A binary rewriter could easily be used to provide a similar defense mechanism as ASLR. An interesting future avenue of research is to investigate software diversity through binary rewriting.

Control Flow Integrity. Control Flow Integrity (CFI) [3] is a basic safety property that can prevent attacks from arbitrarily controlling program behavior. CFI dictates that software execution must follow a path of a control-flow graph that is determined ahead of time by analysis (in this case, static binary analysis is performed). CFI is enforced using static verification and binary rewriting (with Microsoft's Vulcan [28] tool) that

instruments software with runtime checks. These checks aim to ensure that control flow remains within a given control-flow-graph. CFI is a very effective defense against buffer overflow attacks (and any attack which attempts to change a program's control flow) since any attempt by an attacker to divert the control flow of a program will be caught by CFI. However, the main barrier to CFI's adoption seems to be the overhead associated with the scheme. The average overhead of CFI in the prototype implementation is 16% on the SPEC2000 benchmarks. Also, unlike SecondWrite, the binary rewriter used by CFI depends on a binary being compiled with debug information which is usually not available in production binaries. If a binary is not compiled with debug information then CFI cannot be currently applied.

Our schemes implemented through our binary rewriter can provide the same level of protection as CFI. An additional advantage of our scheme is that our binary rewriter does not require access to any special information in an input binary unlike all previous binary rewriters (including the binary rewriter used in CFI) which require access to relocation or debug information.

Program Shepherding. Program Shepherding [17] employs an efficient dynamic software machine-code interpreter (DynamoRIO [6]) for implementing a security enforcement mechanism. A broad class of security policies can be implemented using a machine interpreter such as DynamoRIO. For example, DynamoRIO could be used to enforce control-flow integrity. Program shepherding enforces a similar policy that imposes certain runtime restrictions on control flow such that an attacker can not alter a program's flow of control.

Program Shepherding can experience significant memory and runtime overheads, particularly on the Windows platform. The scheme requires an application and interpreter to be run simultaneously. The high overheads of interpretation in some cases are likely to limit adoption of Program Shepherding. Further, unlike using off-line rewriters like SecondWrite, Program Shepherding requires the installation of an extra piece of heavyweight software (DynamoRIO) in addition to the application to be run.

2.3 Related Work in Binary Rewriting

Binary rewriting and link time optimizers have been considered by a number of researchers. Binary rewriting research is being carried out in two directions: static rewriting and dynamic rewriting. Dynamic binary rewriters rewrite the binary during its execution. Examples are PIN [19], BIRD [20], DynInst [13], DynamoRIO [6], Valgrind [21], and the translation phase of VMWare [2]. Dynamic rewriters are hobbled since they do not have enough time to perform complex compiler transformations; they have been primarily used for code instrumentation and simple security checks in the past. Moreover dynamic rewriters do not have the time to perform deep code analysis needed to discover program features needed for static optimization of security checks. Finally dynamic rewriters encounter run-time overhead from the act of rewriting, which can be substantial. Given these drawbacks, we do not discuss dynamic rewriters further.

The methods in this research are primarily directed at static binary rewriters such as our rewriter, SecondWrite. Existing static binary rewriters include Etch [23], ATOM [11], PLTO [24], Diablo [29], and Vulcan [28]. Three points of novelty for our work are as follows. First, we are not aware of any rewriter adding our particular set of existing

compile-time security schemes to binaries. Second, none of the existing rewriters employ a compiler level intermediate representation; rather they define their own low-level machine-code-like custom intermediate representation. This has several downsides: (i) most existing rewriters cannot modify the stack layout since they do not distinguish individual objects on the stack. Hence they cannot implement security schemes that modify the stack; and (ii) most existing rewriters recognize functions, but not their arguments or return values, and hence cannot deploy security schemes that employ these schemes. SecondWrite overcomes both these problems as we will describe in section 4.

A third point of novelty of our work is that all existing rewriters can only rewrite binaries that contain relocation or debug information. This information, present at link-time, is usually discarded in COTS binaries for two reasons – it is not needed for execution; and vendors legitimately fear it can be used to reverse engineer their binaries. Indeed of twenty commercial and open-source binaries we surveyed, *none contained either relocation or debug information*. As a result, existing binary rewriters would not be able to rewrite those binaries at all. In effect, existing binary rewriters can only be deployed by developers, not end-users. In contrast our rewriter (SecondWrite) can rewrite arbitrary binaries even without relocation or debug information, as we will describe in section 4. This renders our platform a uniquely powerful tool for allowing anyone to rewrite binaries from any source to enable any security scheme they want.

3 Background on Binary Rewriting

This section presents some background on binary rewriting and discusses how security enforcement interacts with it. Our approach relies on innovative binary rewriting schemes [26, 4] incorporated into our binary rewriting infrastructure called SecondWrite. Binary rewriters are pieces of software that accept a binary executable program as input, and produce an improved executable as output. The output executable typically has the same functionality as the input, but is improved in one or more metrics, such as run-time, energy use, memory use, security or reliability.

Advantages of binary rewriting. In recognition of its potential, binary rewriting has seen much active research over the last decade. The reason for great interest in this area is that binary rewriting offers additional advantages over compiler-produced optimized binaries:

– **Ability to do inter-procedural optimization.** Although compilers in theory can do whole-program optimizations, the reality is that they do little if any. Many commercial compilers - even highly optimizing ones - limit themselves to separate compilation, where each file (and sometimes each function) is compiled in isolation. In contrast, binary rewriters have access to the complete application all at once, including libraries. This allows them to perform aggressive whole-program optimizations to exceed the performance of even optimized code. This ability can be useful for security schemes as well; in particular for those schemes that rely on whole-program information such as call graphs and inter-procedural properties to either work at all, or to optimize fully.

- **Ability to do optimizations missed by the compiler.** Some binaries, especially legacy binaries or those compiled with inferior older compilers, often miss certain optimizations. Binary rewriters can perform these optimizations missed by the compiler while preserving the optimizations the compiler did perform. This property may help the rewriter overcome some of the overheads of security enforcement by improvements in program run-time.
- **Increased economic feasibility.** It is cheaper to implement a code transformation once for an instruction set in a binary rewriter, rather than repeatedly for each compiler for the instruction set. For example, the ARM instruction set has over 30 compilers available for it, and the x86 has a similarly large number of compilers from different vendors and for different source languages. The high expense of repeated compiler implementation often cannot be supported by a small fraction of the demand. This implement-once property is useful for security schemes as well.
- **Portable to any source language and any compiler.** A binary rewriter works for code produced from any source language by any compiler. This is a significant advantage for a security scheme such as the one presented in this paper. A scheme would not need to be ported to various compilers but would instead only need to be implemented once within a binary rewriter. Portability of rewriters aids security schemes implemented in them as well.
- **Works for hand-coded assembly routines.** Code transformations cannot be applied by a compiler to hand-coded assembly routines, since they are never compiled. In contrast, a binary rewriter can transform such routines. Applying security in a binary rewriter has the advantage of working for hand-coded assembly versus compiler implementation of security, which does not.

Architecture of Binary Rewriter. The binary rewriter developed by our group and utilized for this research is named SecondWrite. Figure 1 presents an overview of the SecondWrite system. SecondWrite's custom binary reader and de-compiler modules translate the input x86 binary into the intermediate representation (IR) of the LLVM compiler. LLVM is a well-known open-source compiler [18] developed at the University of Illinois, and is now maintained by Apple Inc. LLVM IR is language- and

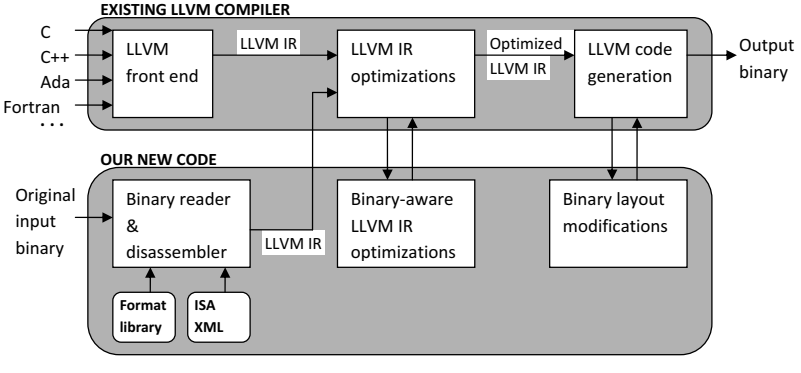


Fig. 1. SecondWrite system

machine-independent. Thereafter the LLVM IR produced is optimized using LLVM's pre-existing optimizations, as well as our enhancements, including security enforcement in this paper. Finally, the LLVM IR is code generated to output x86 code using LLVM's existing x86 code generator.

The front-end module consists of a disassembler and a custom binary reader which processes the individual instructions and generates an initial LLVM IR. This module reads the format of instructions from Instruction Set Architecture (ISA) XML files for the ISA in question, allowing for targeting of the rewriter to different ISAs. Currently SecondWrite rewrites x86 and ARM binaries. To give an idea of the effort needed for retargeting, consider that the sizes of the x86 and ARM XMLs are approximately 14000 and 1500 lines of code (LOC), respectively. The XML for x86 is much larger since it is a complex CISC ISA whereas ARM is RISC. This is a relatively small portion of the total size of SecondWrite, which exceeds 120,000 LOC (mostly C++). From this we can see the effort required for retargeting to a new RISC ISA is relatively modest (1-2 person-months in our estimate).

4 Innovations in SecondWrite

SecondWrite has three innovations that make it especially powerful, and a good platform for security enforcement. To be practical for security enforcement, a rewriter must satisfy three requirements. First, it must be able to rewrite stripped binaries (*i.e.*, those without relocation information) since most real-world binaries are stripped. Second, it must be able to rewrite the entire code, not just discoverable parts of it, thus achieving 100% code coverage. Third, it should rewrite the code to high-level IR, since some security schemes rely on high-level constructs such as functions, arguments, return values, and symbols. Below we describe why existing static rewriters do not provide any of these three capabilities, but SecondWrite does. We note that SecondWrite (and any similar tool) does not work with software that is either self-modifying or performs integrity self-checks.

Rewriting without relocation information. A key innovation in SecondWrite is that it can rewrite stripped binaries, *i.e.*, those without relocation or symbolic information, unlike existing rewriters such as ATOM [11], PLTO [24], Diablo [29], and Vulcan [28] which cannot. Relocation information is generated by the compiler to help the linker in resolving addresses that can change when files are linked. Symbolic information may be inserted for debugging. However, production binaries almost never contain such information since linkers delete relocation information by default. The programmer may instruct the linker to retain such information. However corporations almost never release binaries with relocation and symbolic information since they are unnecessary for execution, and they fear such information can be used to reverse-engineer information about their code.

The requirement for relocation information in existing rewriters arises from the need to update the target addresses of control-transfer instructions (CTIs) such as branches and calls. When rewriting binaries, code may move to new locations because instructions may be added, deleted or changed compared to the original code. Hence the targets of CTIs must be changed to their new locations. Doing so is easy for direct CTIs, since their targets are available in the CTI itself; the target can be changed to its new

address in the output binary. However for indirect CTIs, the target may be computed many instructions before at an *address creation point* (ACP). It is impossible to find all possible ACPs for each CTI using dataflow analysis since they may be in different functions and/or propagated through memory (memory is not tracked by dataflow analysis.) Hence existing rewriters require relocation information to identify all possible ACPs. All ACPs must be present in relocation information since ACPs are precisely the list of addresses that need relocation during linking.

SecondWrite has devised technologies to rewrite binaries without relocation information. Details are in [26]; here we briefly summarize the intuition of our method. Rather than trying to discover ACPs, our basic method relies on inserting run-time checks at indirect CTIs that translate the old target to its corresponding new address using metadata tables that store such translations for all possible old branch and call targets. Aggressive alias analysis on the indirect CTI target is used to prune the list of such possible targets to a small number. Further, compile-time optimizations are applied when possible to reduce the number of run-time checks. The result is a method than can rewrite arbitrary binaries without relocation or symbolic information with very low overhead. The rewriter can then perform security enforcement on arbitrary binaries for the first time.

Achieving 100% speculative code coverage. A key challenge in binary rewriters is discovering which parts of the code section in the input binary are definitely code, and thus should be rewritten. This is complicated since code sections often contain embedded data such as literal tables and jump tables which if rewritten by mistake will result in an incorrect program. The only way to be sure a portion of the code section is indeed code is to find a control-flow path from the entry point of execution to that portion. However portions of code may be reachable only through indirect control-transfer instructions (CTIs). Unfortunately the precise value set of CTI targets cannot be discovered statically in all cases; hence not all code may be discovered. Existing rewriters may not discover all the code, yielding incomplete code coverage – undiscovered code cannot be rewritten, and thus security cannot be enforced on it.

SecondWrite overcomes this problem by speculatively rewriting portions of the code segment which cannot be determined to be surely code, thus achieving 100% speculative code coverage. The detailed scheme is in [26]; but the intuition is that portions of the code segment which cannot be proven to be code are speculatively disassembled as if they are code anyway. If the speculative code turns out to indeed be code at run-time, then it is executed, achieving 100% speculative code coverage. Instead, if the speculative code arose from disassembling data bytes, that incorrect speculative code will never be executed since control will never transfer to it at run-time; preserving correctness. Instead the data is accessed from a copy of the original binary maintained in the rewritten binary. Maintaining this code copy increases code-size, but not the I-cache footprint since only the data portions of it are actually accessed, thus run-time is not affected. Since machines today have vastly more resources than even a few years ago, an increase in code size without increasing run-time is tolerable, especially given the payoff of being able to rewrite any binary.

Rewriting to high-level intermediate representation (IR). Unlike SecondWrite which represents programs in the high-level compiler IR, existing rewriters represent the binary using binary-like low-level code in the rewriter, making the program harder to analyze and modify. For example, high-level program features required for some security schemes, such as function arguments and return values, are not apparent in the binary. Further, existing rewriters retain register and memory accesses as-is, unlike SecondWrite which replaces both by symbolic accesses. Having memory accesses is problematic since it forces the layout of memory to be retained exactly in the rewritten binary, preventing modifications and optimizations of the stack and global segments, and additions to the stack segment. This too is inconvenient for security check insertion since such checks may allocate their own stack memory in some cases.

SecondWrite overcomes these programs by representing the binary code internally in compiler IR. Our method, described in [4], relies primarily on two technologies. First, high-level program features such as functions, and their arguments and return values are discovered from the binary using deep static analysis. Second, registers and memory locations are replaced by symbols as in high-level programs, allowing easy compiler modification of the memory allocation. With the resulting high-level IR, security checks become easy to apply.

5 Methods

One of the contributions of this paper is that we have carefully chosen a set of complementary and effective schemes that, taken together, achieve the goal of defending against all types of buffer overflow attacks at the lowest combined run-time cost. The totality of our schemes protect against not only the commonly known stack buffer overflow into return addresses, but is much more general than that, in that they protect against buffers on the global, stack and heap segments from overflowing onto a variety of code pointer locations that are possible in any data segment, including return addresses, function pointers, indirect branch pointers, *longjmp* buffers, and base pointers[3].

We implement our scheme by adding various passes that operate on high-level IR inside our binary rewriter. Our overall scheme consists of a number of components that we describe in detail in this section.

Stack Canary Insertion. The first component of our scheme is the simplest. LLVM provides the ability to insert stack canaries during code generation. Utilizing this capability allows us to provide nearly the same level of protection to an un-protected binary as StackGuard [8] would provide when given an application's source code.

Essentially, a random canary value is generated at run-time and placed on the stack during a function's prologue. In the function epilogue, the value stored on the stack is compared with the random canary value for this process. If there is any difference, execution is halted as the canary value has been corrupted.

Base Pointer Elimination. The *old base pointer* which resides on the stack is a data pointer that points to the base of the parent function's stack frame. Compilers sometimes introduce it since it makes it convenient to restore the stack pointer at the end of the function and to

[3] Base pointers are not code pointers but lead to a similar vulnerability [31].

address different stack locations with the same offset even as the stack grows and shrinks in the function. When it is present in the input binary, it introduces a vulnerability just as dangerous as a code pointer [31]. This is because the old base pointer can be attacked by building a fake stack frame with a return address pointing to attack code, followed by overflowing the buffer to overwrite the old base pointer with the address of this fake stack frame. Upon return, control will be passed to the fake stack frame which immediately returns again redirecting flow of control to the attack code.

Given our unique use of LLVM IR in SecondWrite, the elimination of the base pointer in the output binary becomes a simple matter even when the input binary has base pointers. LLVM is an optimizing compiler and the binaries produced by LLVM are highly optimized. One common optimization applied by modern compilers on the x86 platform is to free up the EBP register for register allocation by removing the base (or frame) pointer. We used this LLVM pass to eliminate the base pointer from the binary.

When the base pointer is eliminated by LLVM, any attack relying on overwriting the base pointer is immediately prevented. There will be no base pointer for an attacker to modify. While corruption of the stack may still occur if an attacker overflows a buffer in order to attempt to overwrite the base pointer, no attack will be successful.

Return Address Protection. Given that stack canaries as inserted by LLVM do not provide the same level of protection as the ProPolice mechanism that comes with GCC, we decided to implement a more complete solution similar to the protection scheme in StackShield [30], that protects against corruption of the return address. The basic idea of our return address protection scheme is as follows:

1. During the function prologue, push the return address of the current function in a return address stack implemented in a global data structure. For multi-threaded applications, multiple "shadow" stacks are maintained.
2. In function epilogue, compare the current return address on the stack with the value popped from the top of the return address stack.
3. If there is any difference between these values, execution is halted.

This simple scheme will detect if the return address has been modified either directly or indirectly. We implemented this scheme as it is relatively simple and protects against both direct and in-direct modifications of the return address. It also requires no modification of the stack layout and prevents modifications of the return address through buffer overflows in the heap or global segments.

Two challenges with this scheme are as follows. First, its overhead might be significant since every function has an associated security overhead incurred every time it is called. We found this overhead to be especially significant for recursive functions since they tend to short-running. Second, the size of the return address stack might be significant for deeply nested recursive functions, and we would have to bound it a-priori, which is hard to do.

We applied an optimization for relieving this problem which we call the *return address check optimization*. We observed that this protection mechanism is only necessary if a function contains a write to a stack buffer since return addresses only exist on the stack. This is hard to determine without symbolic information, so we conservatively try to prove that a function only has directly addressed memory references to constant

addresses. If it finds any indexed write (base + runtime-variant offset), then it conservatively assumes that it could be a buffer write, and disables the optimization. If all writes are provably non-indexed writes to a constant offset, it enables the optimization, *i.e.*, the protection mechanism is turned off in the function. Thus the optimization saves on run-time overhead without sacrificing any protection.

We found this optimization surprisingly effective since it works best for small leaf functions in the call graph, and for recursive functions, which happen to be precisely the functions dynamically called most frequently. During our experimental evaluation of our scheme, of the many recursive functions we found, every one of them had its check optimized away. This is unsurprising since recursive functions tend to be short running, and unlikely to allocate stack arrays (although they may access portions of global arrays, such as in quicksort, but those still are optimized.) As a result of the optimization, the run-time overhead for scheme is greatly reduced, and the required return address stack depth is also greatly reduced. Of course, the overflow of the return address stack is not an error as we add extensions to it on the heap upon overflow, which slows execution, but is extremely rarely invoked even for small return address stack sizes of (say) 256 addresses.

Function Pointer Protection. One common attack method used by attackers is to overwrite a function pointer so that when it is de-referenced, code of the attacker's choosing will be executed. In a binary executable, function pointers will appear as indirect calls. Thus, another component of our scheme concentrates on protecting all indirect calls and branches similar to how function pointers are protected in StackShield [30].

Our scheme adds checking code before all indirect calls and branches. A global variable is declared at the beginning of the data segment and its address is used as a boundary value. The checks inserted before any indirect call or branch ensure that the target of the indirect call or branch points to memory below the address of the global boundary variable. If the target points above the address of this global boundary variable then execution is halted.

An assumption in the above scheme is that a process follows the standard UNIX layout with the data segment above the code segment. This scheme does not protect against return-to-libc attacks since the target of the indirect call will still be within the code segment.

Protection for longjmp *buffers.* The paired functions *setjmp* and *longjmp*, present in most C and C++ libraries, provide a means to alter a program's control flow in addition to the usual subroutine call and return sequence. First, *setjmp* saves the environment of the calling function (say *foo()*)into a data structure, and then *longjmp* in another function (say *bar()*) can use this structure to jump back to the point it was created, at the *setjmp* call. As a result, execution will return from *bar()* to *foo()* even when *foo()* is not the immediate parent of *bar()*. A typical use for *setjmp*/*longjmp* is exception handling.

The data structure used by *setjmp* for saving the execution state is referred to as a *jmp_buf*. Within this structure, enough information is stored to restore a calling environment. In particular, one member of this structure saves the value of the program counter which is used when restoring the calling environment. An attack method used by attackers is to overwrite the value of the program counter stored in the *jmp_buf* structure after a call to *setjmp* and before a call to *longjmp*. If this happens, control will be transferred

to an address of the attacker's choosing when the *longjmp* is executed. Our method for defending against attacks of this kind is as follows:

1. Create a hash table within the global segment of the rewritten binary. Protect the hash table with write-protected (via *mprotect()*) guard pages, to mitigate attacks against it.
2. After each call to *setjmp* store the current value of the program counter in the *jmp_buf* structure into the hash table.
3. Before a call to *longjmp* get the current value of the *jmp_buf* structure that will be used. Attempt to perform a lookup in the hash table for the value of the program counter.
4. If the lookup in the hash table fails, then the value of the program counter has been modified and so we abort; otherwise execution continues

We expect the run-time overhead of this scheme to be very low in practice, since *setjmp* and *longjmp* calls are very rare. To the best of our knowledge, this scheme is the first protection scheme designed to protect against longjmp buffer attacks in the manner described. We intend to extend our scheme to cover the *ucontext_t* buffers and the *getcontext(), setcontext(), swapcontext()* API that is meant eventually to replace the *setjmp/longjmp* API.

6 Experimental Evaluation

We now present and discuss experimental results from our evaluation of our system. First, we examine the effectiveness of our security schemes as implemented in SecondWrite on a set of security benchmarks previously proposed by Wilander and Kamkar [31] for evaluating the effectiveness of buffer overflow defenses. Second, we examine how effective our scheme is in protecting against real-world attacks on widely-used real code (not benchmarks). Third, we examine the overheads of both the binary rewriter and our security scheme on some SPEC2006 and other benchmarks.

Synthetic Results. In order to test how effective our scheme is, we utilized the benchmarks provided by Wilander and Kamkar [31]. Twenty buffer overflow attack forms were developed, in order to evaluate the effectiveness of tools available at the time that aimed to mitigate buffer overflow attacks. The attack forms covered every combination of buffer overflow attacks on global, stack, and heap buffers overflowing to a return addresses, base pointers, function pointers, and *longjmp* buffers. An attack form is defined as a combination of a technique, location, and an attack target. Of the twenty attack forms, we obtained the source code to only eighteen of these (*i.e.,* the other two were not available to us for evaluation). We then compiled the programs into binary code which we then rewrote using SecondWrite. Our schemes in SecondWrite successfully defended against all attack forms in the Wilander and Kamkar benchmarks.

Real World Attacks. Ultimately, the success of our scheme depends on whether or not attacks that are observed in the real world can be prevented or not. Two real-world attacks were tested.

The first application we tested was GHTTPD – an HTTP server. This web server has a stack buffer overflow vulnerability in its logging function [15]. We obtained an exploit for GHTTPD which overflows a stack-based buffer and corrupts the return address. Using the return address protection component of our scheme, we were able to protect the return address and prevent the attack that uses the buffer overflow vulnerability to corrupt the return address. When our scheme is enabled, the return address corruption is detected when the attack occurs and the application is aborted.

The second application we tested was another HTTP server named CoreHTTP. This application contains a buffer-overflow vulnerability where it fails to adequately check user-supplied data before copying it to an insufficiently sized buffer [14]. We obtained an exploit for this application and applied our protection scheme to the application. Again, when our protection scheme is enabled, the attack is detected and the application is aborted.

Binary Rewriting Overhead. A subset of SPEC benchmarks and other benchmarks were selected to substantiate the performance of our binary rewriter. The benchmarks were selected at random, and are limited only by the criteria that they are correctly rewritten by our still-early prototype. Table 1 lists the set of benchmarks that are used in the experiments. All the benchmarks are compiled with gcc 4.4.1. At this point, Second-write is not mature enough to rewrite large real-world commercial applications which are hence not included; debugging is ongoing. There are no fundamental limitations we know of in rewriting such programs.

Table 1. Application Characteristics

Application	Source	Lines of C Source Code
lbm	SpecFP2006	1155
art	OMP2001	1914
mcf	SpecInt2006	2685
libquantum	SpecInt2006	4357
sjeng	SpecInt2006	13847
hmmer	SpecInt2006	35992
h264	SpecInt2006	51578

In the first experiment, all binaries executed correctly after rewriting thus demonstrating SecondWrite's robustness. The standard suite of LLVM optimization passes ran without any changes in SecondWrite. These include CFG simplification, global optimization, global dead-code elimination, inter-procedural constant propagation, instruction combining, condition propagation, tail-call elimination, induction variable simplification and selective loop unrolling.

Besides correctness, the next most important metrics are the run-time speedup or overhead of the rewritten binaries versus the input binaries. For this paper, we study the performance of our rewriter on already optimized binaries. Figure 2 shows the normalized execution time of each rewritten binary compared to an input binary produced using GCC with the highest available level of optimization (-O3 flag). The results are

mixed, with most benchmarks nearly breaking even or showing a small slowdown, one benchmark showing a larger slowdown of 20%, and one benchmark actually shows a speedup of 16%. The average is 2.7% slowdown.[4]

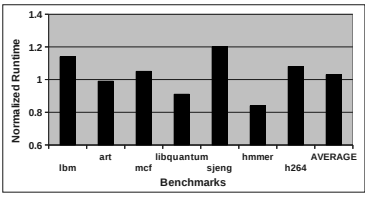


Fig. 2. Normalized runtime of rewritten binary as compared to optimized input binary (runtime=1.0)

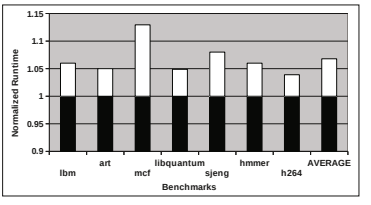


Fig. 3. Normalized runtime of rewritten binary with security scheme added

We consider this near break-even performance on highly optimized binaries a good result for three reasons:

- Our initial goal was not necessarily to get a speedup, but to generate correct IR without without introducing too much overhead. This would enable the IR to be a starting point for various custom compiler transformations we wanted to perform thereafter, such as automatic parallelization or security as covered in this paper. Ultimately, these optimizations determine the utility of the rewriter.
- These numbers represent our first-cut implementation devoid of any attempt at producing a better IR more geared towards optimization. We believe these numbers can be substantially improved with more detailed IR and are exploring several related avenues.
- We have currently not implemented any custom serial optimizations that might improve performance further, such as the inter-procedural versions of common sub-expression elimination and loop-invariant code motion, changing the compiler-enforced calling convention for registers for better run-time, and more aggressive inlining. We believe these optimizations hold promise as the inter-procedural optimization abilities of current compilers are very limited compared to their intra-procedural performance.

One additional advantage of the binary rewriter is that it accumulates optimizations across two compilers—rewritten binaries have an optimization if it is either present in the compiler that produced the input binary, or in the rewriter. In our case, if either GCC or LLVM had an optimization, the output binary should have it. This is why, for example, one of our rewritten binaries (*hmmer*) had a 16% speedup versus the input binary. Although GCC with the -O3 optimization flag is known to produce good code, in some cases it missed promoting structure fields to registers whereas LLVM did, explaining the speedup in *hmmer*. With better IR and more aggressive optimizations, we expect to see more consistent speedups in output binaries in the future.

[4] Rewriting unoptimized input binaries produced using GCC -O0 yields an average speedup of 27% using SecondWrite (not shown) due to its optimizations.

Security Related Overheads. The overhead of the security schemes was measured on the same applications as used for measuring the overhead of the binary rewriter. The results are presented in Figure 3 and show overhead versus rewritten binaries without security schemes inserted. As seen, the average run-time overhead of 6.7% introduced by the protection scheme is low.

7 Conclusions

We have presented a new mechanism using an advanced binary rewriter that allows end users to retrofit powerful security features into third-party, binary-only software. The particular mechanisms we used are well known, and some have been partially implemented in other tools. Our system will allow end-users to retrofit program-level security protections for the first time in a highly customizable manner according to their needs and environment.

We demonstrated the effectiveness of our mechanism via experimental evaluation, begining with the benchmarks developed by Wilander and Kamkar. We successfully mitigated all the attack forms in the benchmarks. We then went on to demonstrate how our mechanism successfully defends against multiple real-world attacks. We also measured the overheads of our binary rewriter in isolation and then we showed what the overhead of adding the security mechanism to a binary is. In both cases, we demonstrated that the overhead introduced is quite low.

Future work involves extending the binary rewriter to work on more substantial applications and demonstrating that the mechanism defends against more real-world attacks and to better handle multi-threaded code and the new *ucontext_t* API. Other interesting avenues for future research are software diversification and self-healing techniques using the binary rewriter we have developed.

Acknowledgements. This work was supported by the Air Force, DARPA and the NSF through Contracts AFRL-FA8650-10-C7024, AFOSR-MURI-FA9550-07-1-0527, DARPA-FA8750-10-2-0253 and NSF-CNS-09-14845, respectively. Any opinions, findings, conclusions or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the US Government, the Air Force, DARPA, or the NSF.

References

1. Smashing the stack for fun and profit. Phrack magazine 7(49) (1996)
2. List of VMWare White Papers, http://communities.vmware.com/docs/DOC2601
3. Abadi, M., Budiu, M., Erlingsson, U., Jigatti, J.: Control-flow integrity. In: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS), pp. 340–353. ACM, New York (2005)
4. Anand, K., Smithson, M., Kotha, A., Elwazeer, K., Barua, R.: Decompilation to Compiler High IR in a Binary Rewriter. Tech. rep., University of Maryland (November 2010), http://www.ece.umd.edu/~barua/high-IR-technical-report10.pdf
5. Boyd, S., Kc, G., Locasto, M., Keromytis, A., Prevelakis, V.: On The General Applicability of Instruction-Set Randomization. IEEE Transactions on Dependable and Secure Computing (TDSC) 7(3) (July-September 2010)

6. Bruening, D.: Efficient, transparent, and comprehensive runtime code manipulation. Ph.D. thesis (2004)
7. Cowan, C., Beattie, S., Johansen, J., Wagle, P.: PointGuardTM: Protecting pointers from buffer overflow vulnerabilities. In: Proceedings of the 12th Usenix Security Symposium (2003)
8. Cowan, C., Pu, C., Maier, D., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q., Hinton, H.: StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In: Proceedings of the 7th USENIX Security Symposium, pp. 63–78. USENIX Association (1998)
9. Cowan, C., Wagle, P., Pu, C., Beattie, S., Walpole, J.: Buffer overflows: Attacks and defenses for the vulnerability of the decade. In: Proceedings of DARPA DISCEX, p. 1119. IEEE Computer Society, Los Alamitos (2000)
10. Eto, H., Yoda, K.: propolice: Improved Stack-smashing Attack Detection. Transactions of Information Processing Society of Japan 43(12), 4034–4041 (2002)
11. Eustace, A., Srivastava, A.: Atom: a flexible interface for building high performance program analysis tools. In: Proceedings of the USENIX Technical Conference, pp. 25–25 (1995)
12. Foster, J.: Buffer Overflow Attacks: Detect, Exploit, Prevent. Syngress Media Inc. (2005)
13. Hollingsworth, J.K., Miller, B.P., Cargille, J.: Dynamic program instrumentation for scalable performance tools. In: Proceedings of the Scalable High-Performance Computing Conference, pp. 841–850 (1994)
14. CoreHTTP Http.C Buffer Overflow Vulnerability, http://www.securityfocus.com/bid/25120/info
15. ghttpd log() Function Buffer Overflow Vulnerability, http://www.securityfocus.com/bid/5960/info
16. Hu, W., Hiser, J., Williams, D., Filipi, A., Davidson, J., Evans, D., Knight, J., Nguyen-Tuong, A., Rowanhill, J.: Secure and practical defense against code-injection attacks using software dynamic translation. In: Proceedings of the USENIX Conference on Virtual Execution Environments (VEE) (2006)
17. Kiriansky, V., Bruening, D., Amarasinghe, S.: Secure execution via program shepherding. In: Proceedings of the 7th USENIX Security Symposium (2002)
18. Lattner, C., Adve, V.: LLVM: A compilation framework for lifelong program analysis & transformation. In: Proceedings of the International Symposium on Code Generation and Optimization (GCO), pp. 75–87 (2004)
19. Luk, C.K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V.J., Hazelwood, K.: PIN: Building Customized Program Analysis Tools with Dynamic Instrumentation. In: Proceedings of the ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI). pp. 190–200 (2005)
20. Nanda, S., Li, W., Lam, L.C., Chiueh, T.: BIRD: Binary Interpretation using Runtime Disassembly. In: Proceedings of the International Symposium on Code Generation and Optimization (CGO), pp. 358–370 (2006)
21. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. ACM SIGPLAN Notices 42(6) (2007)
22. Rescorla, E.: Security Holes...Who Cares? In: Proceedings of the 12th USENIX Security Symposium, pp. 75–90 (August 2003)
23. Romer, T., Voelker, G., Lee, D., Wolman, A., Wong, W., Levy, H., Bershad, B., Chen, B.: Instrumentation and optimization of Win32/Intel executables using Etch. In: Proceedings of the USENIX Windows NT Workshop on The USENIX Windows NT Workshop (1997)
24. Schwarz, B., Debray, S., Andrews, G., Legendre, M.: Plto: A link-time optimizer for the Intel IA-32 architecture. In: Proceedings of the Workshop on Binary Translation (WBT) (2001)

25. Shacham, H., Page, M., Pfaff, B., Goh, E., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. In: Proceedings of the 11th ACM conference on Computer and Communications Security (CCS), pp. 298–307 (2004)
26. Smithson, M., Anand, K., Kotha, A., Elwazeer, K., Giles, N., Barua, R.: Binary Rewriting without Relocation Information. Tech. rep., University of Maryland (November 2010), http://www.ece.umd.edu/ barua/without-relocation-technical-report10.pdf
27. Solar Designer: "return-to-libc" attack. Bugtraq Mailing List (August 1997)
28. Srivastava, A., Edwards, A., Vo, H.: Vulcan: Binary transformation in a distributed environment. Tech. Rep. MSR-TR-2001-50, Microsoft Research (2001)
29. Van Put, L., Chanet, D., De Bus, B., De Sutter, B., De Bosschere, K.: Diablo: a reliable, retargetable and extensible link-time rewriting framework. In: Proceedings of the IEEE International Symposium On Signal Processing And Information Technology, pp. 7–12 (December 2005)
30. Vendicator: Stack shield technical info file v0.7. (2001), http://www.angelfire.com/sk/stackshield/
31. Wilander, J., Kamkar, M.: A comparison of publicly available tools for dynamic buffer overflow prevention. In: Proceedings of the 10th Network and Distributed System Security Symposium, pp. 149–162 (2003)
32. Witten, B., Landwehr, C., Caloyannides, M.: Does open source improve system security? IEEE Software 18(5), 57–61 (2001)

Generating Optimised and Formally Checked Packet Parsing Code

Sebastien Mondet, Ion Alberdi, and Thomas Plagemann

University of Oslo, Norway
{smondet,plageman}@ifi.uio.no, ion.alberdi.research@gmail.com

Abstract. While implementing distributed applications, the parsing of binary packets is a very difficult and error-prone task the developer has to face. Moreover, these programming mistakes are often the source of distant vulnerabilities. In this paper we present a code-generation library, called Promiwag, for creating optimised and safe packet parsing code. Its input is concise human-readable descriptions of the protocols and the interests of the application in specific pieces of information. Promiwag follows a dependency-based algorithm, and uses high-level optimisation techniques to generate minimal parsing automatons. These automatons can be compiled into C or OCaml code for efficient execution, and to annotated Why code. This latter output is then used to automatically prove that for any possible input packet, the generated code cannot perform any illegal memory access, and that no infinite loop can be triggered. We have used our code generator to implement a pretty-printer for Internet protocols, and we provide experimental results on the performance of the generated code.

1 Introduction

One of the currently observable trends for future distributed applications is the convergence of Internet and Mobile networks. More and more small, mobile, and cheap electronic devices must be able to communicate with personal laptops, servers, and/or *clouds*. Therefore, one needs to implement specialised and complex software for many different platforms. Some of these platforms require special attention to be given to performance. Indeed, computing resources may be very limited, e.g., on mobile devices. In parallel, the error-proneness and the lack of static guaranties of the common tools and languages used by developers (unsafe memory accesses, dynamic typing, etc.) lead to a very large amount of bugs and security vulnerabilities present in distributed applications[1].

One common task that many distributed application developers have to handle is the parsing of binary protocols. For example, this is the case in almost every piece of protection software (firewalls, intrusion detection systems, etc.). The code handling these aspects, especially when hand-written in C or C++, is very often swarming with implementation mistakes. The latter often lead to exploitable buffer over-flow vulnerabilities or easy-to-trigger denial-of-service attacks [18].

[1] c.f. CVE (Common Vulnerabilities and Exposures); cve.mitre.org

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 173–184, 2011.

Code generation techniques (a.k.a. *Meta-Programming*) have already been often used to automate programming tasks. Examples range from the `lex` and `yacc` tools developed in the 70s, to the compiler of the "Fastest Fourier Transform in the West" [11]. Generating code from a higher level representation allows adapting the output specifically to the target platform. It also allows providing aggressively optimised code while keeping the required level of safety. From the initial input representation, one can also generate additional output which may assist the developer in other ways; e.g. for testing or documentation extraction.

The safety provided by code generation will often be much better than with hand-written code. However, the code generator itself is generally a piece of software developed by one or more human beings. As such, it can (and will) be the victim of many programming mistakes. How can one trust the code generated by an external tool? Expert and careful proof-reading of thousands of lines of generated code is by far too expensive for most software projects; and *testing* requires identifying in advance all the possible ways to make a program go wrong. When dealing with input as binary packets which can be erroneous, or maliciously crafted, both approaches are not realistic. We address this issue by mathematically ensuring the safety of the generated code against any possible input. Formal properties are automatically proved on the output of the Promiwag library.

In this paper, we focus on the generation of code for parsing network packets. From a high-level representation of network protocols and their handling by the user, our code generator creates optimised code on which safety and security properties are formally proved. The user describes i) the binary formats of the packets, ii) the transitions between (micro-)protocols (e.g. how to pass from IP to TCP), iii) a few logical properties to insure, iv) which pieces of information he is interested in, and v) what user-function to call with the parsing results as arguments. Out of these concise declarations, the Promiwag library builds a parsing automaton which parses only the necessary fields (i.e. the fields required by the user and their dependencies). This intermediate representation also contains logical properties about the algorithms involved. It is then compiled into i) C code for high performance parsing, ii) OCaml code, and iii) annotated Why code [10]. This latter output allows, thanks to the Alt-Ergo theorem prover [8], automatically proving that, for any possible input packet, no wrong memory accesses can be provoked, and no infinite loops can be triggered.

In Section 2, we detail and discuss the design of our code generation approach. Then, in Section 3, we assess the performance of the code through experiments. We discuss related projects in Section 4, before concluding and suggesting future work (Section 5).

2 Packet Parsing Code Generation

Figure 1 illustrates the high-level idea of the structure of the code generation process in the Promiwag library. We have developed our code generation tool as

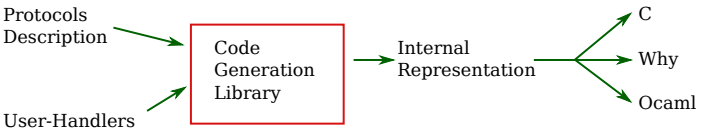


Fig. 1. The Structure of the Code Generation Process in the Promiwag Library

an Objective Caml[2] library. The user can link their programs with the library to generate the code. A dedicated and independent input language could be easily implemented, but the slight benefit of readability would not overcome the loss in flexibility. Indeed, as Promiwag can generate code for different target platforms, and can adapt the code to the needs of the user, having a powerful programming language can be very practical for code reuse and maintainability.

We present the code generation process from an input/output point of view in Sections 2.1 and 2.2. Afterwards, we discuss the limitations and some issues related to the current state of our implementation in Sections 2.3 and 2.4.

2.1 Input

For each protocol or *micro-protocol* (e.g. for a given type of packet header, and its links to the "upper" layers), the input required from the user is composed of two declarative pieces of code; the (micro-)protocol description and the user handler.

The first, is the description of the (micro-)protocol itself. The user gives a precise definition of the binary format, i.e., they name the fields and their size(s). A field can be an integer of any size between 0 and 32 bits, or a string of any byte size. They also provide the "parsing transitions", which are a sequence of *switch-like* statements to tell the generator how to continue the parsing. For example, given the value of a protocol field, the parser has to continue with another (micro-)protocol included in a given payload. Finally, the user can provide "run-time checks" that tell the generator to include assertion checking code, about the fields of the parsed packet. The goal of the run-time checks is not to write a full stateless firewall, but to insure a few consistency properties needed for the formal proofs on the code. For example, the soundness of the binary format may depend on assumptions about certain fields in a given (micro-)protocol.

The second declaration, the *user-handler*, is optional. It describes the user's interest in specific parsable values and their handling by an external function. Each user-handler refers to a described (micro-)protocol. The user specifies their request for values, offsets, pointers, and/or sizes in the binary format, and provides a function in the target programming language (C or OCaml) which expects these values as arguments. The function returns a boolean/integer deciding whether to continue parsing or not.

[2] ocaml.org

```
let ipv4_format = packet_format [
    fixed_int_field "version" 4;
    fixed_int_field "header_length" 4;
    fixed_int_field "tos_precedence" 3;
    fixed_int_field "tos_delay" 1;
    fixed_int_field "tos_throughput" 1;
    (* ... Skipped ...*)
    fixed_int_field "dest" 32;
    string_field "options"
      (size (`align32 (`sub
        (`mul (`var "header_length", `int 4),
         `add (`offset "dest", `int 4)))));
    payload  ~name:"ip_payload"
      ~size:(size (`sub (`var "length",
                `mul (`var "header_length", `int 4))))
  ]
let ipv4_transitions = switch "protocol" [
    (* ... Skipped ...*)
    case_int_value  6  tcp "ip_payload";
    case_int_value 17  udp "ip_payload";
    (* ... Skipped ...*)
    case_int_value 47 gre  "ip_payload";
  ]
let ipv4_checks = [ check_range "header_length" 5 15 ]
```

Listing 1.1. Example of Description of IPv4's Protocol

```
(ipv4_name, [ `value "src"; `value "dest"; `size "ip_payload" ],
 call_target_handler "ipv4_handler_function")
```

Listing 1.2. Example of User-Handler for IPv4

Listings 1.1 and 1.2 show (shortened) examples of user code regarding the IP protocol (version 4). The first is a (micro-)protocol description; the binary format, the parsing transitions, and a single run-time check. The second listing is the corresponding user-handler specification. In the binary format description, one can see how variable-length fields like the "`options`" are created with arithmetic expressions. This particular case requires a run-time check. Indeed, the positiveness of the size of the field depends on the fact that the field "`header_length`" is between 5 and 15 as required by the RFC 791. If the corresponding run-time check is not provided by the user, the formal proving of the parsing code will fail. This non-provability matches a hypothetical malicious attack of crafted IP packets containing wrong "`header_length`" fields. Note that this run-time check is the only one needed to implement and prove the *Tcpdump-lite* experiment in Section 3.

2.2 Output

From the previously described declarations, Promiwag builds the parsing automaton code. It starts from the fields required by the user request, the parsing transitions, and the run-time checks, and then follows their dependencies to minimise the amount of code. This parsing automaton is expressed in an internal representation. This representation contains the parsing algorithm in a strictly-typed imperative language. The language is low-level as it can handle pointer

arithmetic, bit-wise operations, etc., but platform independent (for example, endianness is explicit). It is also *annotated* with logical properties gathered during its construction. Many human-readable comments are also kept for debugging or proof-reading purposes. During the code-generation, the library performs relatively high-level code optimisations, including constant propagation, partial evaluation, common sub-expression elimination, and "factorisation" of buffer access verifications. The lowest-level optimisations are left to the back-end compilers of the output languages; implementing them would be redundant.

In the example presented in Listings 1.1 and 1.2, the initial "required fields" correspond to the ones requested by the user (the values of "`src`" and "`dest`", and the size of the "`ip_payload`"), the field "`header_length`" for the run-time check, the value of "`protocol`" and the offset of the "`ip_payload`". The generator needs the last two to be able to continue the parsing with the next protocol. Then, the computation of, for instance, the length of the "`options`" field will be required as a dependency in order to compute the offset of the payload.

The internal representation of the parsing code can then be "compiled" into the three targets shown in Figure 1. On the one hand, the two "programming language" targets are C and OCaml. The former may be seen as the main target, as the code generator was originally designed for C. The transformation to this language is parametrised by a representation of the target platform (endianness, sizes of the types, etc.). The OCaml output was implemented in order to *show* that adding other target languages is a relatively easy task. Implementing this transformation from scratch took only 4 hours for one developer. After this time, the OCaml version of the *Tcpdump-lite* application presented in Section 3 was successfully tested. Generating OCaml is useful by itself: it is used for some experiments for instance to count the buffer accesses in Table 1.

On the other hand, Promiwag's code generator uses the intermediate representation to generate input for the *Why* tool. *Why*[3] is a "verification condition generator" created by Jean-Christophe Filliâtre. For a given annotated algorithm, it propagates "weakest preconditions" [9] and generates "proof goals" when an assertion needs to be proved. The annotations are, generally, intermediary proof goals or "hints" to *guide* the proving steps. The generated proof goals must then be proved by another tool, an automatic prover and/or a proof assistant.

In our case, we use the "fast" weakest precondition algorithm presented by Barnett and Leino [2]. Thanks to the annotations, the proof goals can be automatically proved by the Alt-Ergo[4] theorem prover [8]. The *Why* code generated by the Promiwag library represents the algorithm of the parsing automaton. The user-handlers for the protocols are ignored; we prove only the code we generate. The accesses to the packet buffer are abstracted as logical functional parameters. They are modelled in order to require a proof that they are not out of bounds each time they are "called/used", and at the same time, to express that accessing a packet can lead to *any* arbitrary value, i.e., there is absolutely no assumption on the content of the packet. The automaton is able to parse "protocol loops",

[3] why.lri.fr
[4] alt-ergo.lri.fr

like IP/GRE/IP, but we provide enough annotations to guide the proof that the parser always strictly "advances" in the packet. Hence, given that a packet has constant arbitrary size, the tools can prove that the parsing always finishes eventually.

At the end of a successful proving process, we can assert that the generated parsing automaton verifies the following theorem:

For any possible input packet,
- **no unsafe memory accesses can happen, and**
- **no infinite loops can be triggered.**

2.3 Current Limitations

As *work-in-progress*, the Promiwag library currently does not implement any ad-hoc "parsing state" management. This means that the generated code cannot *yet* use information from previous packets, and/or from other (micro-)protocols in the same packet. The implementation of features like the handling of IP fragmentation or the transitions present *inside* protocols like DNS or DHCP are subject to future work. They have to be implemented for now within the user's handlers; note that the user can still use generated code for the low-level parsing related to this.

The theorem which can be proved on the code, is certainly a huge benefit over most other implementations of packet parsing; especially from a security standpoint. The proof asserts that the code cannot be used to provoke a fatal error. However, we do not have *yet* any formal proof that the code does what it is actually supposed to do. This means that checking the correctness of the computation of a given field of a (micro-)protocol requires testing. Of course, testing for example that the "header length" field parsed for an IP packet has the expected value is much easier than trying to imagine all the possible ways to alter it in order to make the program crash. Moreover, in C or C++ programs, the error reporting of unsafe memory accesses depends on the exact memory layout at the time of testing. They may be completely silent in some cases and appear later *in production*; this problem is completely addressed by the current proof.

Finally, from an *ergonomics* point of view, the error reporting related to formal proving is still quite difficult to understand. In the example of Listings 1.1 and 1.2, if one forgets the run-time check on the "header length", the theorem prover just fails saying "I don't know" about a possibly very long proof goal. Then, tracking down the error to find the original mistake is not an easy task. For now, we have implemented a very explicit naming of generated variables together with high-level comments which are propagated to the different outputs. While these are helpful, more sophisticated techniques will be needed in the future.

2.4 Trust Issues

When hearing about mechanised and/or automated proofs, one has to ask the question: 'Who and what do we *really* have to trust?' Regarding the theorem proved on the generated code, apart from the soundness of the underlying mathematical framework, one has to trust our work in some respects, and the external tools in others.

On the one hand, the trust *bottleneck* in the Promiwag library is the final *compilation* stage. Currently, there is no proof that the different outputs are semantically equivalent. This means that when the theorem prover succeeds, one has to trust *us* that the C and OCaml outputs are equivalent, and actually *proved* by the proof on the Why code. To address this, we have restricted the size of the critical code to the minimum. We also note that implementing the OCaml output in half a day (c.f. Section 2.2) partially shows that this part of the code is quite straight-forward.

On the other hand, the whole tool-chain depends on many external tools. First, one has to trust the theorem proving tools; Why and Alt-Ergo themselves. Both rely on formally proved algorithms, but their implementations are not. Note that other theorem provers can be used with Why if needed. Second, every software project has to trust its compilers; in our case, the C and OCaml compilers. For C, we use the GNU Compiler Collection (GCC). Even though it is widely tested, at least for the C language within the mainstream architectures, many bugs are often found in official releases. As a side note, we have successfully tested the compilation of the C code generated for our Tcpdump-lite example with Compcert[5]. Compcert is a formally proved compiler for a subset of the C language [13]. For OCaml, we use the standard compilers. They may also be a trust issue but a substantial part of the system (compiler and run-time) has already been qualified under the DO-178B standard to be used as tool for critical aircraft software development [16].

3 Experimental Results

In this section, we provide the results of our experimental study into the performance of the generated code. We aim to show that the use of a code generator does not degrade the performance compared to hand-written code, and that the size of the generated code is still quite controllable by the user.

We have used the Promiwag library to implement a pretty printer for Internet protocols (Ethernet, IPv4, ARP, GRE, TCP, UDP, etc.). As previously mentioned, we have named it "Tcpdump-lite". It is based on libpcap[6], and uses our code generator to call user-handlers which are only pretty-printing the requested values (`printf`). We have generated three different *"flavours"* of Tcmpdump-lite:

- *Full*: requests various meaningful fields for every (micro-)protocol, and displays them.
- *Muted*: requests the same fields as *Full* but does not print them; (micro-)protocol user-handlers are just empty, it just prints "error" messages (i.e. alerts when packets are malformed).
- *Light*: requests and prints fields only for UDP and TCP.

To estimate the running times of the tool-chain, we just measure the times taken by the different programs on an old Pentium 4 desktop computer for the *Full*

[5] compcert.inria.fr
[6] www.tcpdump.org

version: i) compiling a code generation program and linking against our library: 0.12 s; ii) running the code generation: 0.04 s; iii) compiling the C code output with GCC: 0.13 s; iv) running the Why tool: 13.8 s; and v) proving all the goals with Alt-Ergo: 21 s.

We have done experiments listening to *"live"* network interfaces and with many "PCAP capture files" (generated by us or found on the Web). To simplify their presentations, we present here the results obtained with two characteristic files. We call the first one `Fuzz-10K`, it is a capture of 10 000 packets, where many are malformed, and contain potential "attacks". The second one is `24GRE-132` which is composed of 132 packets among which many are using 24 encapsulated GRE tunnels. This means that for those packets the "protocol stack" is: `Ethernet/IPv4/GRE/IPv4/GRE/IPv4/.../GRE/IPv4/UDP` where there are 24 GRE encapsulations. Note that handling multiple encapsulations is an important feature [1] which many networking tools cannot handle; explicitly or not. For example the Section 1.8.1 of the Snort manual [19] states that for more than one GRE tunnel, packets will not be inspected, and the Linux kernel simply crashes for more than 36 encapsulations (c.f. Debian bug number 599816).

The first experiment is an analysis of the control the user has on the generated code. Table 1 shows a comparison between the *Full* and *Light* versions of our Tcpdump-lite. We provide the raw size of the whole program, and the number of buffer accesses actually executed during the parsing of both capture files. This shows that, from the same set of (micro-)protocol descriptions, just by removing requests and user-handlers, the user can lighten the application significantly. It matches the goal of writing software *adapted* to the resources of the target device. For example, one can make a special version of a firewall application for mobile phones with a more affordable development effort.

Table 1. Code Comparison of The *Full* and *Light* Versions

	Size (bytes)		**Buffer Accesses**	
	Binary	C File	`Fuzz-10K`	`24GRE-132`
Full	19 157	35 027	242 050	8 457
Light	14 378	22 272	65 482	2 921

We also provide results for brute performance evaluation. We compare the running times of the different versions of Tcpdump-lite, against an "empty" PCAP application (i.e., the same code as Tcpdump-lite but without any parsing of the packets) and different options of the original Tcpdump application. To be less unfair with the latter, we use the options `-n` so that IP addresses are not converted to host names and `-K` so that checksums are not computed. We also redirected all outputs to `/dev/null`. We measure Tcpdump's (noted "T") running times for the different verbosity levels: the default, `-v`, `-vv`, and `-vvv`. In order to try to reduce the entropy of the measurements, we run them 2000 times on an old machine (*Pentium 4*) and on a more recent one (*Core 2 Duo P8700*).

The results are presented in Table 2. Note that the `Fuzz-10K` capture involves some malformed packets which trigger error-message printing even for the *Muted*

Table 2. Results for 2000 runs (times in seconds)

Machine:	***Pentium 4***		***Core 2 Duo P8700***	
File:	`Fuzz-10K`	`24GRE-132`	`Fuzz-10K`	`24GRE-132`
Empty	10.97	6.88	5.16	2.40
Full	113.35	11.02	84.32	5.64
Muted	13.49	7.36	6.94	2.52
Light	19.71	7.44	11.97	2.61
T	122.38	10.79	86.57	5.86
T -v	168.76	14.67	123.49	8.69
T -vv	169.14	15.16	127.49	9.18
T -vvv	168.46	15.17	127.67	9.20

version, and that the `24GRE-132` one induces a lot of parsing logic compared to the display of information. If we assume the two following approximations: i) the *Empty* program represents the operating system plus the "PCAP machinery", and ii) the difference between *Full* and *Muted* measures the time spent in printing, then we can compute the following statistics (for the *Pentium 4*[7]):

- For the capture `Fuzz-10K`, the distribution of the time spent is: *OS+Pcap*: 9.7 %, *Generated code*: 2.2 %, Printing: 88.1 %; and the average time per packet for the *Full* version is: 5.67 μs.
- For the capture `24GRE-132` the distribution of the time spent is: *OS+Pcap*: 62.4 %, *Generated code*: 4.4 %, Printing: 33.2 %; and the average time per packet for the *Full* version is: 41.7 μs.

The running times for the original Tcpdump are given as a comparative indication. Indeed, the application is more feature-full, and dynamically configurable (i.e. on command line).

4 Related Work

The use of code-generation techniques in order to improve the performance and/or the safety of distributed systems has been gaining a lot of interest in literature. For instance, with the *Statecall Policy Language* presented by Madhavapeddy et al. [14], one can describe statefull automatons which are *compiled* into three different targets: OCaml code to embed in an application, Promela code to check temporal properties with the SPIN[8] model checker, and HTML/Ajax code for real-time monitoring of the application. Another example is the declarative sensor networks (DSN) platform (Chu et al. [7]). There, the user describes application overlays using a declarative language (paradigm similar to Prolog) and

[7] The results for the *Core 2 Duo P8700* are slightly better for the generated code but the *true* parallelism of the architecture may make these computations *less meaningful*.

[8] spinroot.com

the compiler generates code for sensors in NesC (the C dialect for the TinyOS operating system).

Regarding the management of packet streams, two projects use meta-programming for high-level packet filtering purposes. First, the BPF+ packet filter [3] uses a high-level language to describe boolean predicates on packet flows. This language is compiled to bytecode which can be verified or interpreted (or JIT-assembled). Among the verifications, the *halting* of the application is insured by forbidding any kind of cycles or loops in the filtering program. Second, the FFPF project (*Fairly Fast Packet Filter* [5]) implements a filtering language which allows one to "chain" (mostly independent) packet analysis applications while sharing kernel-space buffers to avoid copying.

Examples of projects using code-generation especially for packet parsing are *Binpac*, *GAPAL*, and *Melange's MPL*. Binpac [17] is a C++ code generator for the Bro intrusion detection system[9]. It uses a yacc-like language to mix the specification of binary formats and user's C++ code in order to help them in their task. *State* and parsing transitions are written directly by the user as well as bit-level accesses, e.g. by including inline C++ code. GAPAL [4] is a high-level language for analysing application protocols which is evaluated by a C++ interpreter. The Meta-Packet Language of the Melange framework [15], is a domain specific language to describe Internet protocols. Its compiler generates optimised OCaml pieces of code for each protocol; the use of a memory-safe language naturally avoids wrong buffer accesses. It also features ad-hoc parsing state management for situations commonly found in Internet protocols. Parsing transitions between protocols are left to the developer (but they fit well with ML-style programming).

Even though these projects may handle more complex protocols and/or fragmentation thanks to ad-hoc parsing-state management, they all have a unique output language and they do not give any formal proof on the generated code. Moreover, they all generate code for the whole defined set of protocols, i.e. even for non-used portions. As they rely on a lot of hand-written code, brute performance comparisons would be quite meaningless. For example, to compare the performance of the *Tcpdump-lite*, implemented with Promiwag, against another one implemented with *Binpac*, the developer would have to implement the parsing transitions or the "bit-level" accesses directly in C++. This would mean comparing a lot of generated code against hand-written code.

5 Conclusion

Our goal is to provide reduced development costs while keeping performance and enforcing safety. We have presented a code-generator which, from concise and high-level descriptions, can generate optimised code for parsing binary packets. The code is safe-by-construction thanks to automated formal proofs done on the output. The user control the amount and the performance of the generated code. Indeed, the generation is *dependency-based*; we build only the code that is

[9] www.bro-ids.org

actually needed by the user. The experiments we provide assess that the run-time performance will not be degraded by the use of our library.

Future work will be divided in three directions. First, state, persistence, and memory management are needed to add features to the parsing generation. This problem is much more generic than just packet parsing and will have to be treated as such to be used in other aspects of distributed applications. Second, more aggressive optimisations on the code could be implemented. As the performance of a given program transformation depends on the actual code, potential optimisations could be evaluated on the generated code in order to choose the right *option* at the last time. Finally, the formal proving framework gives us a lot of room for interesting improvements. More properties could be proved on the code, e.g. the accuracy of the computations regarding the user's requests, or the absence of dead code. The "trust bottleneck" of the Promiwag library, i.e. the code which transforms the (last) internal representation to the different targets, could be re-implemented in a *directly* certified way with a proof assistant like Coq[10]. This family of formal tools have "executable code extraction" capabilities, and these kinds of program transformations, even if they require a lot of specification and proving work, are well suited for programming with proof assistants [12,6].

References

1. Alberdi, I., Owezarski, P., Nicomette, V.: Luth: composing and parallelizing midpoint inspection devices. In: NSS 2010: Proceedings of the 4th International Conference on Network and System Security, pp. 9–16. IEEE Computer Society, Melbourne (September 2010)
2. Barnett, M., Leino, K.R.M.: Weakest-precondition of unstructured programs. In: PASTE 2005: Proceedings of the 6th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, pp. 82–87. ACM, New York (2005)
3. Begel, A., McCanne, S., Graham, S.L.: Bpf+: exploiting global data-flow optimization in a generalized packet filter architecture. In: SIGCOMM 1999: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 123–134. ACM, New York (1999)
4. Borisov, N., Brumley, D., Wang, H.J., Dunagan, J., Joshi, P., Guo, C.: Generic application-level protocol analyzer and its language. In: NDSS (2007)
5. Bos, H., de Bruijn, W., Cristea, M., Nguyen, T., Portokalidis, G.: FFPF: Fairly Fast Packet Filters. In: OSDI 2004: Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation (2004)
6. Chlipala, A.: Certified Programming with Dependent Types. Online in-progress textbook (2009)
7. Chu, D., Popa, L., Tavakoli, A., Hellerstein, J., Levis, P., Shenker, S., Stoica, I.: The design and implementation of a declarative sensor network system. In: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (2007)

[10] coq.inria.fr

8. Conchon, S., Contejean, E., Kanig, J., Lescuyer, S.: Lightweight integration of the ergo theorem prover inside a proof assistant. In: AFM 2007: Proceedings of the Second Workshop on Automated Formal Methods, pp. 55–59. ACM, New York (2007)
9. Filliâtre, J.C.: Verification of non-functional programs using interpretations in type theory. J. Funct. Program. 13(4), 709–745 (2003)
10. Filliâtre, J.: Why: A Multi-Language Multi-Prover Verification Tool. Research Report 1366, LRI, Université Paris Sud (2003)
11. Frigo, M.: A fast Fourier transform compiler. ACM SIGPLAN Notices 34(5) (1999)
12. Leroy, X.: Mechanized semantics. In: Logics and Languages for Reliability and Security. NATO Science for Peace and Security Series D: Information and Communication Security, vol. 25, pp. 195–224. IOS Press, Amsterdam
13. Leroy, X.: Formal verification of a realistic compiler. Commun. ACM 52(7), 107–115 (2009)
14. Madhavapeddy, A.: Combining Static Model Checking with Dynamic Enforcement using the Statecall Policy Language. In: International Conference on Formal Engineering Methods (2009)
15. Madhavapeddy, A., Ho, A., Deegan, T., Scott, D., Sohan, R.: Melange: Towards a functional Internet. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems (2007)
16. Pagano, B., Andrieu, O., Moniot, T., Canou, B., Chailloux, E., Wang, P., Manoury, P., Colaço, J.L.: Experience report: using Objective Caml to develop safety-critical embedded tools in a certification framework. In: ICFP 2009: Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming, pp. 215–220. ACM, New York (2009)
17. Pang, R., Paxson, V., Sommer, R., Peterson, L.: binpac: a yacc for writing application protocol parsers. In: IMC 2006: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, pp. 289–300. ACM, New York (2006)
18. SANS Institute: Top 20 internet security problems, threats and risks. section 5 anti-virus software (2007), http://www.sans.org/top20/2007/#s5
19. Snort Team: Snort Users Manual. The official documentation produced by the Snort team at Sourcefire (2010)

Organizational Power and Information Security Rule Compliance

Ella Kolkowska and Gurpreet Dhillon

Swedish Business School, Örebro University, Sweden
School of Business, Virginia Commonwealth University, USA
ella.kolkowska@oru.se, gdhillon@vcu.edu

Abstract. This paper analyzes power relationships and the resulting failure in complying with information security rules. We argue that inability to understand the intricate power relationships in the design and implementation of information security rules leads to a lack of compliance with the intended policy. We conduct the argument through an empirical, qualitative case study set in a Swedish Social Services organization. Our findings suggest a relationship between dimensions of power and information security rules and the impact there might be on compliance behavior. This also helps to improve configuration of security rules through proactive information security management.

Keywords: dimensions of power, information security, security compliance.

1 Introduction

Lack of compliance with security policies occurs because of a number of reasons. Foremost amongst them are the inability of the policy to reflect current practices [1] and stakeholder resistance to security rules [2]. The organizational studies literature has intricately linked the concept of resistance to organizational power [3, 4]. Following on from the arguments presented in the dominant literature, we make a call to better understand organizational power in the context of information security policy compliance. Information security policy consists of a number of rules for protecting information in an organization. Whenever security rules are implemented or modified, there is a resultant organizational change - business processes get re-engineered, reporting structures get modified, technical controls get redesigned. However as Hardy [5] suggests, organizational power provides the energy to realize change. Thus, we argue that by developing a good understanding of organizational power dimensions it would be possible to ensure better security rule compliance. Correspondingly we also argue that a better appreciation for organizational power will ensure correct configuration of security rules.

The objective of the paper is to apply Hardy's dimensions of power to understand compliant and non-compliant behavior. Our findings also suggest how managers can use such an understanding to improve compliance with security rules. Three classes of definitions ensue from our argument - organizational power, information security and compliance. In this paper we refer to organizational power as "the probability within a

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 185–196, 2011.

social relationship of being able to secure one's own ends even against opposition" [6]. We refer to information security as the protection of all information handling activities, may these be technical or non technical [7]. And compliance refers to a "relationship in which an actor behaves in accordance with a directive supported by another actor's power, and to the orientation of the subordinated actor to the power applied" [8 pg3].

2 Security Rule Compliance and Organizational Power

In a seminal paper, Ranson et al [9] while discussing specialization of tasks in organizations, have argued that over time the path of internal differentiation leads to a "process of perpetual fission that fragments the collective enterprise of adequate understanding". This means that over time, in any enterprise, as complexity sets in, organization power is bound to get manifested. Hence compliance with a certain "paradigm or problematics" [cf. 9] is an attempt to articulate the latent relationships amongst stakeholders.

In the context of our research, and in using Ranson et al [9] terminology, a security rule is a form of organizational structure, which has its own "devotees". With time security rules get transformed (i.e structures evolve) as do the "devotees". The constant interplay between the evolving structures and those who believe in them results in *power*, which as Hardy [5] notes, helps in "bringing about strategic action". In the literature this interplay has been termed as structures being "constituted and constitutive" [see 9, 10].

Therefore in this section we explore the relationship between organizational power and security rule compliance. It is important to address this issue since dominant literature illustrates a rather consistent pattern of lack of compliance with security rules [11-14]. The notion of lack of compliance as a consequence of organizational power manifestations has been well documented in the literature. Lapke and Dhillon [2] identified resistance to security policies as one of the major reasons for failure. Lapke and Dhillon [15] also consider the importance of understanding organizational power in formulation and implementation of security policies. While aspects of compliance have been touched upon in the work of Lapke and Dhillon [2, 15], they do not explicitly focus on organizational power and it's utility (or limitation) in security rule compliance.

Another stream of security rule compliance research has focused on sanctions. The emphasis of this stream has been on studying the relationship of penalties and pressures that one party might apply on the other. Organizational power that comes into play in the context of penalties and sanctions is generally coercive in nature [16]. Coercive power and rewards have been extensively researched in the management literature. In the context of information security, a number of researchers have argued that coercion, sanctions and rewards have a significant impact on compliance or non compliance [17, 18]. This orientation has lead to using deterrence theory to suggest that individual expectations about external contingencies (e.g., rewards, punishments, etc.) are a driving force that directs their compliant behaviors [19, 20]. The emphasis within such behavioral research [17] is on the modification of one kind of attribute (value congruence, legitimacy, etc) or another, to ensure compliance [see 21, 22, 23].

Some behavioral studies emphasize the importance of value correspondence and cultivation of a security culture. According to these studies compliance can be improved if employees internalize information security values in their daily work practices [24]. In this way "proper" security behavior will become a natural part of an employees' daily work activities [23, 25]. A similar approach is also suggested in 'awareness studies' [26]. These studies suggest that increased security awareness of employees and educational programs leads to better compliance with information security rules [27, 28].

There is no doubt that compliance with security rules can be achieved by any or all of the above identified approaches and clearly there may be more. However a limited number of current studies in the area emphasize the relationship between compliance behaviors and the sociological constructs such as power, which can be utilized to enforce these behaviors. Our study addresses this gap showing the value of applying the dimensions of power to understand compliant and non-compliant behaviors. This is because it allows for clarity on the nature and scope of exiting domination and how it plays out in the context of a strategic change, particularly when a new security rule gets instituted.

Organizational power and its implications on various aspects of business have been well researched and there are a number of conceptions of power. In recent years the work of Cynthia Hardy has had a profound impact on organizational [see 5] and information systems research [see 29]. From Hardy's [5] perspective, power is defined in neutral terms as a force that affects outcomes and allows beneficial results for all involved actors. She suggests a four dimensional framework that helps in understanding the consequences of organizational power from multiple perspectives. While other conceptions of power, particularly Clegg's [30] Circuits of Power have been widely used in information systems and security research, their discussion and comparison with Hardy's conceptualization goes beyond the scope of the current paper.

3 Theory and Methodology

In this section we present theory and methodology used in the conduct of our research.

3.1 Dimensions of Power

According to Hardy's [5] conceptualization of organizational power, it operates along four dimensions: resources, process, meanings and systems. While such an understanding of power has not been used in the information security literature, Dhillon [29] has articulated the dimensions to address IT implementation issues. The dimensions are discussed below.

Resource based power. Hardy contends that resource based power relates to the control that a given individual, group or a role might have on a range of resources available in an organization. Such accumulation of power results in a "carrot" and "stick" situation, which translates to an ability to offer rewards, punish or impose sanctions. Proponents of resource-based power argue that leveraging such power can result in behavioral modification. Critics however argue that repeated use of resource-based power can be counter-productive.

Process based power. Business processes embody the values and norms of organizations. The power of processes challenges the notion that the decision process is open to participation by all interested parties in an organization. In fact, decision processes may be carefully designed to prevent those without power from gaining power by participating, thus protecting the status quo. Process based power can be changed by creating awareness and by opening up processes to new participants, issues and agendas. Such awareness helps sustain new behavior as long as it remains within existing values and norms.

Meaning based power. Power residing in meanings focuses on preventing conflict (i.e. resistance). Conversely, a lack of appreciation of meaning of action, causes resistance. Proper "indoctrination" of less powerful members in organizational customs and hierarchies leads to unquestioning acceptance of their role in the organization, thus preserving the status quo. Through the symbolic use of icons, rituals and language, change is given a new meaning, making it appear legitimate, desirable, rational or inevitable. Changes in some underlying values and norms may be possible. However changes in behavior are usually difficult.

System based power. Townley [31] succinctly describes system based power as "constituted through correlative elements of power and knowledge" p. 522. Power of a system is intertwined throughout all aspects of an enterprise. It cannot be mobilized without the other three dimensions: resources, processes and meanings. System based power is often taken for granted since it lies in the unconscious acceptance of "way things get done" in an organization. System based power is the backdrop against which decisions get taken.

In this paper we show the value of applying the dimensions of power to understand compliance and non-compliance and also to show how managers can mobilize those power dimensions to improve compliance with a security policy. Our contribution in relation to the earlier studies is thus the normative knowledge about how compliance can be improved by mobilizing suitable power dimensions. In the study we apply Hardy's [5] multidimensional framework of power. The framework was successfully used in earlier studies [29] related to IS implementations. We chose this framework for this study because of its value in realizing strategic changes in organizations. We argue that enforcement of security policy in an organization results in radical change i.e. enforcement of security policy is often met with resistance because it necessitates changes in business logic and existing business practices. Such situations are usually a consequence of a combination of resources certain stakeholders may have access to or a lack of clarity in the formal organizational procedures. As a result, meaning of stakeholder actions often remains ambiguous.

3.2 Research Methodology

The study was conducted via a qualitative case study [32, 33] at one of the Swedish Municipality Social Service Divisions responsible for helping vulnerable children and their families. The case study was divided in two parts. The aim of the first part is to create an understanding about the organization and to identify relevant actor groups for studying power relationships. The second part focuses on finding power dimensions that were utilized to influence employee action, their awareness and values related to information security rules.

Data was collected in two phases. In phase one, interviews were conducted and project documents reviewed. Sixteen group interviews with eight different stakeholder groups (including management, system owners, IT-technicians and five user groups) were conducted. Each group included 5-6 people. The interviews focused on identifying and evaluating information flows as handled by the integrated computer based systems within the organization. In our initial analysis, two stakeholder groups emerged as central to use of power in security policy compliance. These were (a) managers who enforced the new security rules and processes through the information system and (b) care providers who did not comply with these rules.

In phase two, document analysis and in-depth interviews were conducted with two actor groups: managers and care providers. The emphasis was on finding different means of power that managers utilized to enforce new security processes and rules. Interviews with social service care providers were also conducted. These helped in interpreting what the respondents felt regarding their actions, awareness, values and perceptions with respect of security rules and processes. The respondents were chosen from all treatment centers. The number of respondents was not pre-determined. Once a saturation level was achieved, further interviews were not conducted. An interview guide helped in conducting the interviews, which mapped onto areas corresponding to information security rules within social services. This helped us in being comprehensive in our data collection efforts. Suitable probes were used and the data was correlated with informant insights. Each interview lasted approximately 1-2 hours. All interviews were tape recorded and transcribed. The data was also related to Hardy's [5] theory on dimensions of power. Particular attention was placed on identifying different means by which power was utilized. These were then classified as per the dimensions. The impact of power on actions, awareness and values was also interpreted.

4 Analyzing Power Dimensions and Information Security Compliance

In this section we analyze the case from a power perspective to understand how different dimensions of power were mobilized to drive the change of work practices in social services and how the changes resulted in compliant and non-compliant behaviors.

4.1 Case Background

In the information security policy at our case study organization, security was defined as: protection of information and information systems to achieve organizational requirements for availability, integrity, confidentiality and traceability. To meet the requirements for information security the municipality board decided that all actor groups working in the municipality's social services were obligated to use an implemented IS for communication and exchange of information. According to the system owners, the information security rules and legal requirement were implemented in the system. Though all actor groups were obligated to comply with the decision taken by the municipality board, it was noticed that one actor group, the care providers at treatment centers, did not comply with the new rules. This lack of compliance caused

information security problems related to confidentiality, availability, integrity and traceability. As a result managers were concerned about confidentiality of information and privacy of the clients. Other actor groups also could not access the up-to-date information and consequently their job performance suffered. Because of the problem, the organization was also exposed to significant legal consequences, which could potentially have an impact on the viability of the enterprise.

4.2 Power of Resources

To improve information security, the municipality board decided that all actor groups within social services were obligated to use a computer-based information system for communication and exchange of information. The information system was implemented in all divisions of the social services. A special module supporting care provider work processes was included in the system. Significant time was allocated for training users. Resources were also allocated to IT-support teams. This ensured support of new users with respect to the system. Furthermore consultants who would be responsible for training of the users were hired.

After implementation of the computer-based information system, all documentation of social work at the treatment centers was supposed to be done in the system. However the care providers were confused about the processes, goals and requirements related to the new way of documenting. Before implementation of the system, paper-based documentation was mainly used as a means for communication between care providers working at a treatment centre. It was therefore clear as to *what* and *why* with respect to documentation. However following the implementation it was unclear as to what should be documented and to what extent. It was also unclear what the main goal with the documented information was. The system had also some serious deficiencies. For instance templates for some important documents were missing in the system and also the system did not support all work processes at treatment centers forcing users to use other resources such as word (for templates) and USB sticks for exchanging information.

In summary, resource based power utilized by managers partially influenced care providers behaviors related to documentation of social work. Care providers did begin documenting information using the implemented system. And they did find the system easy to use and were very satisfied with the technical support they got. However because of confusion regarding the new processes and the deficiencies in it, the users developed their own routines relating to handling of information. Moreover there were no information security rules that regulated these routines. Consequently confidential information was exchanged with help of insecure portable devices and saved as Word files on local unprotected hard drivers. Same information could be documented at different places (manually and digitally) at the same time with risk for loss of integrity. The information was also registered in the system too seldom and consequently not available for the other actor groups when they needed it.

4.3 Power of Processes

Using the information system to communicate and exchange information meant changes in care provider work processes and responsibilities. Care providers were

used to very detailed, paper-based notes to exchange information within a team and at treatment centers. All care providers were responsible for preparing these notes so that no information would be missing. They also had frequent contact with other actor groups. During these contacts it was possible to explain and clear up eventual misunderstandings, as well as to communicate the interpretive dimension of the work. According to care providers this dimension was very important in their work. At meetings with other stakeholders, care providers were responsible for presenting a rich picture of the situation, while care officers were responsible for choosing the relevant information and registering it in the system. In case of exceptional circumstances the responsible care officer was informed immediately by phone or e-mail and then the situation was discussed.

After the system was implemented the care providers were responsible for choosing the relevant information for other actor groups and for registration of that information in the information system. Awareness about the new processes for documentation was created by training and educational courses. The courses focused on functionality in the system and also explained that the registered information should be short and focus on facts. Although the new awareness was created during the courses, care providers still used the old way for communicating and exchanging educational information. The reason for this was conflict between the new processes and care provider work values. Documentation in the system was considered as limiting because it required formal reporting (only facts), while their work was based on interpretation and observations. For the care providers, integrity and availability of information meant that information was detailed and included both facts and interpretations. These values were impossible to achieve according to the new rules because it was difficult to communicate the emotional and interpretive dimension through the system. In summary, the process-based power utilized by management did create awareness about new processes, however the awareness did not satisfactorily influence care providers behavior because it clashed with their own work values.

4.4 Power of Meaning

Power of meaning was not utilized by management in this case. The implemented information system was supposed to improve information security in the organization by enforcement of embedded information security rules. It was assumed that employees in social services were both aware of and aligned with existing information security rules. There was also a strong security culture amongst care providers at the studied treatment centers. In particular confidentiality and privacy were emphasized as two core values. Care providers pointed out that security awareness was very important in their work. Even prior to the computerized systems, sensitive client information was handled very carefully - paper-based notes were locked in special rooms; the old paper-based notes were destroyed, etc. In spite of high information security awareness, care providers did not usually comply with the security rules, since the rules clashed with their own values related to integrity and availability of information.

4.5 Power of System

Power in the system is considered as the status quo and exists in “taken for granted” values, traditions, cultures and structures [5]. This dimension of power is often

beyond the reach of organizational members. Hence to make a change, managers must utilize the other three dimensions of power: resources, processes and meanings.

In our case study organization, managers wanted to improve information security through the implementation of a computer-based information system. The managers relied mostly on power of resources in trying to change employee behavior. By creating a suitable environment and allocating resources for implementing the system and for training of the employees, managers did succeed to partially change employee behavior. However the ambiguity regarding the new processes and responsibilities created confusion amongst employees. Because of the unclear requirements for the new behavior, it was also impossible to fully deploy the power of resources to direct the behavior to support new processes. While the power of resources did create some awareness about new processes and responsibilities, the training rendered was insufficient to change underlying values concerning communication and exchange of information. Thus the management failed to use power of meanings to change employee underlying values and norms so as to give the processes a new meaning. This resulted in information security goals remaining under achieved.

5 Discussion

Four implications seem to emerge. These are based on our case study data. Space limitations however forbid us from going into sufficient details.

1. It is important to consider that an information security rule might involve strategic changes. As described in the case study section, the social services organization implemented most of the security rules as part of the computer based information systems. While some of the rules already existed in the organization, many new ones were also created. From a system administration perspective, implementing security rules simply amounted to careful design of rules into the computer-based systems and then implementing them in the organization. The ongoing argument in the organization was that if the integrated system were used for communication and exchange of information, it would ensure compliance with the security rules. As one of the administrators succinctly put it:

> Compliance with security rules is really a function of ensuring that **all** [emphasis added] organizational communications and information handling took place through the technical system.

This meant that the organization never saw the need to focus on establishing responsibility structures or establishing process descriptions, particularly when new rules were instituted. In the information security literature such perspectives have been termed as 'technically skewed' for a largely socio-technical problem [34]. In our case study organization, the implementation of the system largely occurred because of the power that resided with the administrators. Hence a combination of resource based and process based power was exerted. While the power ensured the implementation, the system forced differing interpretations of rules across the social services. One of the case-workers noted:

> It is practically impossible to use the system since it does not reflect the way we work. The checks and balances that have been built into the system are not necessarily the way in which any of the case-workers operate.

In dealing with such situations, Hardy [5] suggests that by managing meanings, resources and process, it is possible to "redefine the strategic initiative and the changes on which it hinges, as legitimate" (Pg. S10). This helps in creating awareness about the new structures and processes and hence making it possible to control behavior through the deployment of specific resources.

2. New security rules come embedded with structural changes, which require mobilization of power residing in the systems to ensure success. Implementation of a security rule constitutes significant structural changes. Since such changes are typically institutionalized in the organization, it requires a careful consideration of values, traditions and sub cultures. Failure to do so, results in systematic bypassing of the rules or circumventing controls. It occurs largely because of the "we don't do things in such a way here" attitude. In our case study organization, a social worker noted:

> We have been given the new system to undertake work in an efficient manner. I must say that it is not working. The new access rules mean that we have to wait for approvals through the chain of command. Unfortunately when one is with a client, they have to take decisions instantaneously. In such instances, we simply do not use the system. ....now I know that this can have possible security and compliance ramifications, but at the same time I have a job to do and services to render.

A typical approach of the management is to come in with a heavy hand, thus using either the power by virtue of the resources or their control of the processes. This however can be counterproductive. Hardy [5] suggests that instituting changes of this kind are a laborious task where work needs to be done in establishing proper buy-in. This ensures that there is a gradual shift in the prevalent ways of working.

3. New security rules have the potential to introduce value conflicts. Mobilizing power of meaning is important to avoid such conflicts. Whenever new security rules are implemented, they challenge the conventional interactions amongst organizational stakeholders. This usually has the potential for causing value conflicts. Value conflicts result in misinterpretation of meanings. From a dimension of power perspective, the power that resides in the meanings needs mobilization such that there is correct interpretation of the rules. In the literature, such misinterpretations have been linked to information security problems [see 35]. In our case study as well, the IT staff felt that there seemed to be a lack of common understanding as to how the work needs to be carried out. One IT staffer noted:

> I don't understand this. There are usually no complaints about the system. Everything works. However many people do not seem to use it. The new social workers who use the system seem to come up with interpretations that are either different or in disagreement with the experience of the older social workers.

Various researchers stress that conflict and resistance is a natural consequence of applying power of resources and processes for realization of strategic change.

Consequently managers who want to avoid employee resistance have to engage in power of meanings to legitimize their decisions. Lukes [36] has argued that power of meanings is often used to shape perceptions and cognition so that individuals do not question the status quo. The literature also terms this kind of power as "management of meaning" [37] where an individual may legitimize and de-legitimize so as to accept the viewpoint. Various symbols are typically used in this process - redundancy compensation, consultation, good will etc.

4. Power residing in resources, processes, meanings and systems needs to be mobilized to ensure awareness of values to achieve compliance with information security rules. In the studied case, the management failed to create an understanding of the new security processes and the new security rules. Moreover they did not succeed in changing underlying employee values that would affect change in employee behaviors. Consequently the new rules were not fully accepted and employees did not comply with these rules causing problems related to confidentiality, integrity and availability of information. In response to this problem, the organization got involved with a major awareness campaign, focusing on consequences of non-compliance. However the employees did not receive the awareness campaigns. This resulted in significant resistance among the user cadre. One user noted:

> They were bombarding us with all this awareness literature. They were also threatening us about the consequences of non-compliance. Nobody however focused on the reasons why people were not complying to the security rules.

The above observation by one user is a reflection of the state of affairs. In the literature, researchers have argued for awareness about the values rather than awareness about consequences of non-compliance [38]. Hardy [5] suggests that managers can redefine the strategic initiative by mobilizing power in a coordinated manner so as "to influence actions, awareness and values, and avoid both inertia and confusion" (Pg. S11)

6 Conclusion

In this paper we have evaluated power relationships in a social services organization and analyzed their impact on information security rule compliance. While majority of information security research has focused on overcoming resistance through sanctions, we take the position that a better understanding of power relationships helps overcoming resistance to information security rules and hence improve compliant behavior.

In our case study organization the management had failed to realize their plan to improve information security and had fallen short of improving the structures, processes and values. Problems occurred because of two issues. First, the power residing in the organizational structures was not adequately understood. Second, power was only understood in terms of resources. This meant that majority of power exercise resulted in curbing access to resources. While resource based power may work in many cases, it has to be articulated in light of other kinds of power as well. In summary, the paper offers four key findings: 1) It is important to consider that an information security rule might involve strategic changes; 2) Strategic change

requires mobilization of power of resources, processes and meanings and understanding of power embedded in the existing system; 3) Mobilizing power of meaning is important to avoid value conflicts; 4) All dimensions of power need to be mobilized to change actions, awareness and values. This will help in achieving compliance with information security rules.

References

1. Mattia, A., Dhillon, G.: Applying Double Loop Learning to Interpret Implications for Information Systems Security Design. In: The IEEE Systems, Man & Cybernetics Conference, Washington DC, October 5-8 (2003)
2. Lapke, M., Dhillon, G.: A Semantic Analysis of Security Policy Formulation and Implementation: A Case Study. In: The Americas Conference on Information Systems (AMCIS 2006), Acapulco, Mexico (2006)
3. McFarland, D.A.: Resistance as a Social Drama: A Study of Change-Oriented Encounters. The American Journal of Sociology 109(6), 1249–1318 (2004)
4. Markus, M.L.: Power, politics and MIS implementation. Communications of the ACM 26(6), 430–444 (1983)
5. Hardy, C.: Understanding power: bringing about strategic change. British Journal of Management 7, Special issue, S3–S16 (1996)
6. Parson, T.: The structure of social action. Free Press, New York (1968)
7. Dhillon, G.: Principles of information systems security: text and cases. Wiley Inc., Hoboken (2007)
8. Etzioni, A.: A comparative analysis of complex organizations: On power, involvement, and their correlates. Free Press, New York (1975)
9. Ranson, S., Hinings, B., Royston, G.: The Structuring of Organizational Structures. Administrative Science Quarterly 25(1), 1–17 (1980)
10. Benson, J.K.: Organizations: A Dialectical View. Administrative Science Quarterly 22(1), 1–21 (1977)
11. PWC: Security Breaches Survey 2008. Enterprise and Regulatory Reform (BERR). PricewaterhouseCoopers on behalf of the UK Department of Business (2008)
12. Whitman, M.E., Mattord, H.: Principles of Information Security, 3rd edn. Course Technology, Boston (2008)
13. Nash, K.S. Greenwood, D.: The global state of information security. CIO Magazine (2008)
14. Stanton, J.M., Stam, K.R., Mastrangelo, P., Jolton, J.: Analysis of end user security behaviors. Computers & Security 24(2), 124–133 (2005)
15. Lapke, M. Dhillon, G.: Power relationships in information systems security policy formulation and implementation. In: The 16th Annual European Conference on Information Systems (ECIS 2008), Galway, Ireland (2008)
16. Kim, S.H., Lee, J.: A contingent analysis of the relationship between IS implementation strategies and IS success. Information Processing & Management 27(1), 111–128 (1991)
17. Herath, T., Rao, H.R.: Encouraging information security behaviors in organizations: Role of penalties, pressures and perceived effectiveness. Decision Support Systems 47(2), 154–165 (2009)
18. Kankanhalli, A., Teo, H.H., Tan, B.C., Wei, K.K.: An Integrative Study of Information Systems Security Effectiveness. International Journal of Information Management 23(2), 139–154 (2003)

19. Straub, D.: Effective IS security: an empirical study. Information System Research 1(2), 225–270 (1990)
20. Straub, D., Welke, R.J.: Coping with systems risks: security planning models for management decision making. MIS Quarterly 22(4), 441–469 (1998)
21. Boss, S.R., Kirsch, L.J., Angermeier, I., Shingler, R.A., Boss, R.W.: If someone is watching, I'll do what I'm asked: mandatoriness, control, and information security. European Journal of Information Systems 18, 151–164 (2009)
22. Phanila, S., Siponen, M., Mahmood, A.: Employees' Behavior towards IS Security Policy Compliance. In 40th Annual Hawaii International Conference on System Sciences (HICSS 2007) (2007)
23. Thomson, K.L., von Solms, R., Louw, L.: Cultivating an organizational information security culture. Computer Fraud and Security (10), 7–11 (2006)
24. Thomson, K.L.: Information Security Conscience: a precondition to an Information Security Culture. In: 8th Annual Security Conference, Las Vegas, NV, USA, April 15-16 (2009)
25. Vroom, C., von Solms, R.: Towards information security behavioural compliance. Computers & Security 23(3), 191–198 (2004)
26. Puhakainen, P.: A Design Theory for Information Security Awareness. University of Oulu, Oulu (2006)
27. Siponen, M.: A Conceptual Foundation for Organizational Information Security Awareness. Information Management & Computer Security 8(1), 31–41 (2000)
28. Furnell, S.M., Gennatou, M., Dowland, P.S.: A prototype tool for information security awareness and training. Logistics Information Management 15(5), 352–357 (2002)
29. Dhillon, G.: Dimensions of power and IS implementation. Information & Management 41, 635–644 (2004)
30. Clegg, S.: Frameworks of power. Sage Publications, London (1989)
31. Townley, B.: Foucault, power/knowledge and its relevance for Human Resource Management. Academy of Management Review 18(3), 518–545 (1993)
32. Benbasat, I., Goldstein, D.K., Mead, M.: The case research strategy in studies of information systems. MIS Quarterly 11(3), 369–388 (1987)
33. Myers, M.D.: Qualitative research in business & management. Sage Publications, London (2009)
34. Hedström, K., Dhillon, G., Karlsson, F.: Using Actor Network Theory to Understand Information Security Management. In: The 25th Annual IFIP TC 11, Brisbane, Australia, September 20-23 (2010)
35. Dhillon, G.: Managing Information System Security. Macmillan, London (1997)
36. Lukes, S.: Power: a radical view. Macmillan, London (1974)
37. Pettigrew, A.M.: On studying organizational cultures. Administrative Science Quarterly 24, 570–581 (1979)
38. von Solms, R., von Solms, B.: From policies to culture. Computers & Security 23(4), 275–279 (2004)

Delegation of Obligations and Responsibility

Meriam Ben Ghorbel-Talbi[1], Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], Daniel Le Métayer[2], and Guillaume Piolle[3]

[1] Institut TELECOM/Télécom Bretagne
2, rue de la Châtaigneraie, 35576 Cesson-Sévigné Cedex, France
{meriam.benghorbel,frederic.cuppens,nora.cuppens}@telecom-bretagne.eu
[2] INRIA Rhône-Alpes
Inovallée, 655 avenue de l'Europe, 38334 Saint-Ismier Cedex, France
daniel.le-metayer@inria.fr
[3] Supélec
Avenue de la Boulaie, CS 47601, 35576 Cesson-Sévigné Cedex, France
guillaume.piolle@supelec.fr

Abstract. In this paper, we discuss the issue of responsibilities related to the fulfillment and the violation of obligations. We propose to formally define the different aspects of responsibility, namely causal responsibility, functional responsibility, liability as well as sanctions, and to examine how delegation influences these concepts. Our main aim is to identify the responsibility of each agent that is involved in the delegation of obligations. More precisely, we try to answer to the following questions: who is responsible for the obligation fulfillment? When a violation occurs, which agents are causally responsible for this violation? Who is liable for this violation and to whom? And finally, who must be sanctioned?

Keywords: Responsibility, Obligations, Delegation.

1 Introduction

Obligations are important means to specify security control, in particular usage control [14,15,3]. Obligations must usually be fulfilled by a fixed deadline, otherwise violations occur and punitive sanctions are inflicted upon agents (for instance through the activation of prohibitions or new obligations). Yet, agents can violate their obligations due to various causes that can be related to agents themselves (e.g. lack of time or competence), or to other agents who have performed (or not) actions such that they have blocked out the fulfillment of the obligation, or finally to system faults, such as a system dysfunctioning or insufficient authorization/resource [10]. For these reasons, it is necessary to have means to clearly identify the responsibility of agents that are involved in the obligation violation, especially when obligations are delegated to one or more other agents.

Indeed, identifying the responsibility of agents in the case of violations is a fundamental part of security and is central to the determination of liability and sanctions. For this purpose, we focus here on these two issues and we propose,

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 197–209, 2011.

in section 2, a formal model that defines different levels of responsibilities [4], namely functional responsibility which is the operational aspect of an obligation, causal responsibility which expresses the link of causality between an agent's actions and a given fact, and liability which is related to the notion of blame, sanction or damage reparation. In section 3, we propose a model of the concept of obligation delegation. We examine how to deal with the different kinds of responsibilities, and we give a concrete example to illustrate our approach. We give, in section 4, a discussion on related work and concluding remarks.

2 Logical Model of Obligation and Responsibility

In the following, propositions will be noted by lower case italic letters (a, b, p ...), variables by roman strings starting with a capital letter (Var) and litterals by strings in fixed width font (`litt`). Indifferent variables are noted with an underscore (_), using Prolog-like notation.

Basic structure of the model. Our framework is based on the notion of organization. Organizations will be noted $a, b, c \ldots \in \mathbb{O}$. They do not have any kind of property, but we will introduce a way to define arbitrary binary relations between them. This allows to nest organizations, to define roles and other high-level concepts. We choose to represent agents and organizations at the same level, by considering that an agent is itself an organization. Another core component is the notion of obligation. We consider that obligations always come from a normative source (noted $x \in \mathbb{X}$), which is an object shared by a set of organizations. It can be a contract, an order, a law or any kind of normative document. For instance, an organization can publish internal regulations, or several organizations can agree on a contract. Normative sources will be used as references for obligations and associated concepts. The logic does not make any distinction between obligations to be and obligations to do, nor between actions, events and states. These distinctions are abstracted away by the notion of fact (noted $p \in \mathbb{P}$). A fact is a proposition describing a situation or an action. It can be an observation of the system or the object of an obligation. In the remaining of section 2, we will consider a simple obligation (*i.e.* without any delegation), between two agents (or organizations) a and b, coming from a normative source x. We will present the various constructs related to obligations and responsibility, before discussing the impact of obligation delegation in section 3.

Obligations and organizations. Obligation is represented by the modality class O, differentiated in a four-parameter predicate. $O(a, p, b, x)$ represents the fact that a has the obligation, towards b, to ensure p, and that this obligation comes from the normative source x. As our goal is to model the various kind of underlying responsibilities in a fine-grained way, the obligation modality has been emptied of most of its usual meaning, and is best described as the representation of a speech act, the acknowledgment that an obligation has been expressed. Formally, each tuple (O, a, b, x) is a monadic obligation modality, applied to facts. They are defined like in SDL, with a KD axiomatics [21]. The abstract

relation structure is brought by a *relation* predicate. $relation(\texttt{relationName}, a, b)$ means that a is in relation `relationName` with b. Binary relations on $\mathbb{O} \times \mathbb{O}$ can be introduced this way. For instance, $relation(\texttt{playsRole}, a, r, b)$ can mean that a plays a given role r in the organization b.

Functional responsibility is the operational aspect of an obligation, the fact that the obligated agent is actually expected to perform a task itself. We note $FR(a, p, b, x)$ the fact that a has the functional responsibility, for which it is accountable to b, to ensure p, and that this responsibility comes from normative source x. In simple cases, functional responsibility is directly derived from the expressed obligation: if an agent is obliged to ensure p then it has the corresponding functional responsibility. This is why, in this simple (delegation-free) version of the framework, functional responsibility is formally equivalent to obligation (eq. 1). The predicate is introduced to make a distinction between the responsibility and the mere speech act.

$$FR(a, p, b, x) \stackrel{def}{=} O(a, p, b, x) \tag{1}$$

Causal responsibility is not necessarily derived from an obligation, but will contribute to the definition of more complex notions. It expresses the link of causality between an agent's actions and a fact, without any assumption of any kind of "fault". We note $CR_a\ p$ the fact that agent a is causally responsible for the fact p. It implies p itself. It means that a has contributed, in some way to the fact that p is true: there is a causality link between a's behaviour and p. It does not mean that a is the sole responsible agent for p.

We choose to distinguish "material causal responsibility" ($MCR_a\ p$) from "causal responsibility by direct influence" ($CRDI_a\ p$). The former means that p occured, and that there is a causality link between a's actions or inaction and the fact p. The latter means that a made another agent or organization b do something (by the means of an obligation) which made b causally responsible for the fact p. In other words, a used its influence to cause p. Material causal responsibility can be more precisely specified in many ways, by introducing complex relations between the actions and their results. In the version of the formalism presented here however, the notion remains abstract and the individual actions are hidden, because our only need here is to decide wether the causal link exists or not. In the context of this presentation, MCR_a will therefore be considered a primary operator. Yet, we should keep in mind that it is possible to distinguish between various grades of material causal responsibility, which might lead to various grades of other kinds of responsibility. Causal responsibility by direct influence, on the other hand, is defined on the basis of an obligation and of the material causal responsibility of another agent or organization (2). To conclude, this first version of causal responsibility is simply the disjunction of material causal responsibility and causal responsibility by direct influence (3).

$$CRDI_a\ p \stackrel{def}{=} O(b, p, a, x) \wedge MCR_b\ p \tag{2}$$

$$CR_a\ p \stackrel{def}{=} MCR_a\ p \vee CRDI_a\ p \tag{3}$$

Liability. We understand liability with respect to an undesirable fact as the possibility, for an agent or an organization, to be blamed for the fact, to be imposed a sanction. This notion is inspired from the legal concept of liability as it appears in the French legal context, for instance, where a person is held liable if its (faulty) behaviour is causally related to a damage (to another agent or to society). In our model, the damage is represented by a fact p, the fault by a violated interdiction on p and the causal relation by our dedicated operator. It means that if an agent has not violated any norm, then it cannot be blamed or sanctioned. Therefore it may be considered that the system contains very general norms, such as the obligation not to cause a harm or loss to another agent. In our language, $L(a, p, b, x)$ means that a is liable for p towards b, because of obligations coming from normative source x. In a first version its direct form ($DL'(a, p, b, x)$), it is defined as the conjunction between a causal responsibility and a violated obligation (4).

$$DL'(a, p, b, x) \stackrel{def}{=} CR_a\ p \wedge O(a, \neg p, b, x) \tag{4}$$

This direct liability is personal in essence, but in some cases one may be liable for somebody else's actions. For instance, parents often bear civil liability in the name of their children. We need to take this kind of relationship into account, because it can also occur in many organizations, where employers, under certain circumstances, may be liable instead of their employees. In order to model this, we will use a relation `accountableFor`, which we need to be built-in. In short, if a is accountable for b, then we consider that a is liable when b should be. This allows us to define indirect liability $IL(a, p, b, x)$ as in (5). Overall liability (6) is therefore the disjunction between direct and indirect liability, where direct liability DL is redefined as the conjunction between DL' and the absence of a relation `accountableFor`.

$$IL(a, p, b, x) \stackrel{def}{=} CR_c\ p \wedge O(c, \neg p, b, x) \wedge relation(\texttt{accountableFor}, a, c) \tag{5}$$

$$L(a, p, b, x) \stackrel{def}{=} DL(a, p, b, x) \vee IL(a, p, b, x) \tag{6}$$

If different levels of causal responsibility are defined, then different levels of liability will arise. For instance, one can imagine a weaker causal responsibility CR^1 (denoting a partial responsibility) and a stronger one CR^2 (denoting a full, exclusive responsibility). CR^1 and CR^2 could give rise to two levels of liability L^1 and L^2. In some context, a L^1 liability could be considered too weak to give rise to a sanction, while an agent with L^2 liability would be considered "blamable". For simplicity, we will work only with one kind of causal responsibility here. However, several existing propositions could be useful in designing a gradation of causal responsibility, like constructions based on Pörn's D and D' modalities [16], and in particular the recent proposal by Marek Sergot [19].

Sanction. As mentioned above, in the case of obligation violation an agent or an organization has to make good for this violation. We use the predicate $sanction(s, c, p, x)$ to say that sanction s is associated to fact p by normative

source x and may be imposed by agent c. Note that we use this predicate to define sanctions in the sense of punishment (penal responsibility), but also to define blame and the reparation of damage or loss (civil responsibility). We choose not to formally differentiate the two notions. In our language, sanctions are associated to the agent liable for the violation according to the normative source. $S(a, p, b, s, c)$ means that sanction s can be imposed by agent c to a following fact p, for which a is liable towards b:

$$S(a, p, b, s, c) \stackrel{def}{=} L(a, p, b, x) \wedge sanction(s, c, p, x) \tag{7}$$

Some discussion remarks. One question that remains to be answered, for the sanction to be just: is the agent actually able to fulfill the obligation or to avoid its violation? For this purpose, one has to define the concept of the agents' *ability* [5,12] to fulfill a given obligation, as well as the parameters influencing this ability. Thus, when a violation occurs we can tell whether a liable agent was actually able to fulfill the obligation in that moment. Many concepts have to be defined and considered to define agents' ability. For instance, is the agent considered able to do some task if it is able to delegate it to another agent?

Another issue is the possibility of sanctions for agents which are causally responsible for the violation, but which bear no liability with respect to the current source of norms. For instance, organization b may deem agent a liable for a given violation with respect to a source of norms x, but agent c, belonging to a foreign organization on which b has no influence, may have a greater causal responsibility because it prevented a from doing its job properly. No liability of c towards b can apparently be derived, because c is not concerned by x and therefore it has not violated any norm of x. Yet, it would seem just that c could be blamed. No liability can be built upon x, but there may be other applicable normative sources. On the first hand, if a and c share a source forbidding an agent to harm another in the way c did, then a liability can be derived from that, and the corresponding sanction will be considered independently from x. It can also be the case that c broke one of its own norms and is sanctionned for that [10], but that its liability is not towards b. On the other hand, if c has not violated any norm applying to it, then it is not faulty in any way and has neither to be sanctionned nor to provide a reparation. In other words, an agent with no functional responsibility for a given fact cannot be judged liable and therefore cannot be blamed. It matches real world situations, in which a fault must be exhibited for a sanction to be applied. For instance, two shops operating in the same street may have a negative impact on each other's income, thus generating a damage, but as long as none of them breaks the general rules of commerce, no civil reparation or penal sanction can be sought.

3 Modelling Obligation Delegation

Now that the notions of obligation, causal responsibility, functional responsibility and liability are available, we will propose a model of the concept of obligation

delegation and examine its influence on the former notions. We say that an obligated agent b delegates its obligation to another agent a when b obliges a to what b was initially obliged. Depending on the options of this delegation, this may or may not influence the functional responsibility and the liability of both a and b with respect to the obligated fact.

The delegation predicate. The delegation of an obligation is represented by an instance of the *delOb* predicate. $delOb(a, p, b, c, x, \text{FRoption}, \text{Loption})$ means that b delegates to a the obligation on p that it had towards c, coming from the normative source x. The last two parameters are the options of the delegation related to functional responsibility and liability. Functional responsibility can be either shared (FRoption = `fr_share`) or forwarded (`fr_forward`). In the first case, both a and b have functional responsibility: they are both in charge of ensuring p. This is for instance the case if the obligation delegation is a request for help on a complex task. In the second case, a alone gets functional responsibility. b does not have to take actions anymore, it is a's role to actually ensure p. It is not possible that b keeps functional responsibility for itself alone, as the key idea about delegation is giving someone else something to do.

Liability can be kept (Loption = `l_keep`), shared (`l_share`) or forwarded (`l_forward`). If liability is kept, then the delegatee will accept no other liability than towards the delegator. It means that if b is liable towards c and delegates to a with `l_keep`, then a will not be liable to c, only b will. On the other hand, a will still be locally liable to b: it is a way to acknowledge that the speech act of delegation itself generates its own liability. If liability is shared, then both a and b will be liable to c (and a will still be "locally" liable to b). If liability is forwarded, then only b will be liable to c (and to a, locally).

For instance, in a conference program committee a reviewer a can delegate the obligation to review a given paper to an external reviewer b, using options `fr_forward` and `l_keep`. In this case, b has the functional responsibility to review the paper, a is liable to the PC chair if the review deadline is not met, and b is liable to a. We can also imagine the opposite situation: a PhD student delegates the obligation to review a paper to his/her advisor (obviously with his/her consent) using options `fr_share` and `l_forward`. In this case, the student transfers the obligation, i.e. he/she is no more liable to the PC chair, but will help his/her professor to review the paper. Note that there is a hierarchical authority between the professor and the student, therefore the student must request the consent of the professor before delegating the obligation (see [2] for more details about consent negotiation).

Formally, to be valid a delegation from a delegator b to a delegatee a on p necessitates the existence of a prior obligation $O(b, p, c, x)$ (i.e. an obligation to b towards another agent c), and it creates a new obligation for the delegatee a towards b. Note that this obligation is also coming from the same normative source x. Equation (8) illustrates this derivation mechanism. The prior formula says that B delegates to A its obligation on P, coming from source X, with

options LRoption and Loption. The derived formula is the new obligation of A, towards B, to ensure P according to normative source X.

$$\frac{delOb(\mathrm{A},\mathrm{P},\mathrm{B},\_,\mathrm{X},\mathrm{FRoption},\mathrm{Loption})}{O(\mathrm{A},\mathrm{P},\mathrm{B},\mathrm{X})} \tag{8}$$

Rights system. Depending on the context of an obligation, it is not always possible or desirable to delegate it. The initial obligator may demand that the initial obligee keeps either liability or full functional responsibility, for instance. It is therefore necessary to install a rights system over obligation delegation: each normative source will also enacts a number of rights formulae, and depending on the active rules, a specific delegation will be authorized or not. More details about how to set up such contextual rights about delegation are given in [1]. It is currently assumed that normative sources properly define these rights, in that rights enacted by a given source should not interfer with the obligations coming from another one (see [8] for more details about conflict management).

Rights enacted by a normative sources are represented by *allow* and *deny* predicates. $allow(a,b,p,c,x,\mathrm{FRoption},\mathrm{Loption},\mathrm{Recursivity})$ means that an obligation $O(a,p,c,x)$ can be delegated to b with the options FRoption and Loption. If Recursivity = `recursive`, then this permission propagates to any delegated obligation. It does not if Recursivity = `nonrecursive` $deny(a,b,p,c,x,\mathrm{FRoption},\mathrm{Loption},\mathrm{Recursive})$ means that this same initial obligation cannot be delegated with these options. If *deny* is recursive, it means that any delegated obligation is also subject to it. It can be relevant, for instance, if another set of options is authorized for the delegation. Recursivity in the rights system is defined by (9).

$$\begin{aligned} &deny(a,b,p,c,x,\mathrm{FRoption},\mathrm{Loption},\texttt{recursive}) \\ &\quad\rightarrow deny(b,\_,p,a,x,\mathrm{FRoption},\mathrm{Loption},\texttt{recursive}) \\ &allow(a,b,p,c,x,\mathrm{FRoption},\mathrm{Loption},\texttt{recursive}) \\ &\quad\rightarrow allow(b,\_,p,a,x,\mathrm{FRoption},\mathrm{Loption},\texttt{recursive}) \end{aligned} \tag{9}$$

When an agent delegates an obligation, it does not directly instantiate *delOb*, but rather creates an instance of a *delObAttempt* predicate, which generates the corresponding *delOb* only if the delegation is valid according to the existing rights. It should be noted that *allow*s and *deny*s are terms which can be more or less instantiated (parameters can be ground litterals or uninstantiated variables), and thus more or less specific. By default, any delegation that is not allowed is forbidden, and *deny* has priority over *allow*. The overall rule for deriving an obligation delegation from a delegation attempt is described by (10).

$$\frac{\begin{array}{c} delObAttempt(\mathrm{A},\mathrm{P},\mathrm{B},\mathrm{C},\mathrm{X},\mathrm{FRoption},\ \mathrm{Loption}), \\ O(\mathrm{B},\mathrm{P},\mathrm{C},\mathrm{X}),\ allow(\mathrm{B},\mathrm{A},\mathrm{P},\mathrm{C},\mathrm{X},\mathrm{FRoption},\mathrm{Loption},\_), \\ \neg deny(\mathrm{B},\mathrm{A},\mathrm{P},\mathrm{C},\mathrm{X},\mathrm{FRoption},\mathrm{Loption},\_) \end{array}}{delOb(\mathrm{A},\mathrm{P},\mathrm{B},\mathrm{C},\mathrm{X},\mathrm{FRoption},\mathrm{Loption})} \tag{10}$$

Obligation chains. We have seen that an obligation can be delegated with or without delegating (or sharing) liability towards the original obligator.

In order to decide whether an agent is liable towards another for a given obligation, one must know whether there is a chain of obligations (including both the initial one and the delegated ones) between them, and whether liability has been shared or forwarded at each step. This is what the *obChain* predicate does. $obChain(a, p, b, x, \text{L\_chain})$ means that there is a chain of obligations between b (obligator) and a (obligatee) about p, coming from the normative source x. The last parameter can be `l_propagated`, if liability has been kept (so that a may be liable to b), or `l_lost` if liability has been lost somewhere between a and b. This predicate is a convenience abbreviation defined as (11).

$$\begin{array}{l} obChain(a,p,b,x,\texttt{l\_propagated}) \stackrel{def}{=} \\ \quad \left\{ \begin{array}{l} O(a,p,b,x) \\ \vee \left(\begin{array}{l} obChain(c,p,b,x,\texttt{l\_propagated}) \\ \wedge \ \left(delOb(a,p,c,\_,x,\_,\texttt{l\_share}) \vee delOb(a,p,c,\_,x,\_,\texttt{l\_forward}) \right) \end{array} \right) \end{array} \right. \\ obChain(a,p,b,x,\texttt{l\_lost}) \stackrel{def}{=} \\ \quad \left\{ \left(\begin{array}{l} obChain(c,p,b,x,\texttt{l\_lost}) \\ \wedge \ delOb(a,p,c,\_,x,\_,\_) \end{array} \right) \vee \left(\begin{array}{l} obChain(c,p,b,x,\texttt{l\_propagated}) \\ \wedge \ delOb(a,p,c,\_,x,\_,\texttt{l\_keep}) \end{array} \right) \right. \end{array} \tag{11}$$

Functional responsibility (with delegation). Functional responsibility must be redefined in order to take obligation delegation into account. Now an agent or organization has functional responsibility for p if it is obliged to ensure p, but only if that it has not delegated this obligation with the `fr_forward` option (12).

$$FR(a,p,b,x) \stackrel{def}{=} obChain(a,p,b,x,\_) \wedge \neg delOb(\_,p,a,x,\texttt{fr\_forward},\_) \tag{12}$$

Causal responsibility by indirect influence (with delegation). Causal responsibility by influence is the only component of causal responsibility which is related to obligations, so it is the only one we need to reconsider in the light of obligation delegation. So far, we have only defined causal responsibility by direct influence, when the agent we have ordered to ensure p is itself materially responsible for it. We introduce causal responsibility by indirect influence, which captures the fact that this obligation can be further delegated. $CRII_a\ p$ (reading "a is causally responsible, by indirect influence, for p") means that there is a chain of delegated obligations on p between a and some agent b, that b is materially responsible for p, and that this is not a causal responsibility by direct influence (13). Causal responsibility by direct influence and by indirect influence are then grouped in a same "causal responsibility by influence" $CRI_a\ p$ (14).

$$CRII_a\ p \stackrel{def}{=} \neg CRDI_a\ p \wedge obChain(b,p,a,\_,\_,\_) \wedge MCR_b\ p \tag{13}$$

$$CRI_a\ p \stackrel{def}{=} CRDI_a\ p \vee CRII_a\ p \tag{14}$$

It can be interesting to introduce a variant operator: causal responsibility by primitive influence $CRPI_a\ p$, meaning that the issued obligation has not been inherited by delegation (15). Overall causal responsibility is then redefined as the disjunction between material causal responsibility and causal responsibility by influence (16).

$$CRPI_a\ p \stackrel{def}{=} CRI_a\ p \wedge \neg delOb(a, p, \_, \_, \_, \_) \tag{15}$$

$$CR_a\ p \stackrel{def}{=} RCM_a\ p \vee CRI_a\ p \tag{16}$$

Liability (with delegation). The last notion to be redefined is liability, for which the *obChain* predicate has been specially tailored. An agent or organization a is directly liable for p towards b if and only if a is causally responsible for p, there is an obligation chain propagating liability from b to a, this liability has not been lost by delegation and no other agent is accountable for a (17). Indirect liability (18) can be defined in the same way, with overall liability remaining the disjunction of direct and indirect liability.

$$DL(a, p, b, x) \stackrel{def}{=} CR_a\ p \wedge PDL(a, p, b, x) \tag{17}$$

$$IL(a, p, b, x) \stackrel{def}{=} CR_c\ p \wedge PIL(a, p, b, x) \tag{18}$$

Concrete Example [1]**.** Let us assume that agent a has the obligation, towards b, to fulfill p, and a is allowed to delegate this obligation with recursive option (figure 1). If the obligation to ensure p is violated then we have to identify agents that are responsible of this violation, namely, functional responsibility and liability (which is derived from causal responsibility). As shown in figure 1, agent a delegates the obligation to c and shares both the functional responsibility and the liability towards b, so we can derive that $FR(a, p, b, x)$ and $L(a, p, b, x)$. Agent c delegates the obligation to d with `fr_share` option and keeps the liability, so d has the functional responsibility towards b. Then, c delegates the obligation to e and forwards both the functional responsibility and the liability. Therefore, c is no more responsible towards b. Finally, agent e forwards the functional responsibility to f. Thus, e is liable for p and f has the functional responsibility towards b. To summarize, if the obligation $O(a, p, b, x)$ is violated then we have $FR(act, p, b, x)$, for act in $\{a, d, f\}$, and $L(act', p, b, x)$ for act' in $\{a, e\}$. Moreover, as mentioned above, agents are also "locally" responsible towards the agent who delegated to them the obligation (directly or indirectly), but only if they have not forwarded this responsibility to another agent. This is why, agent d has the functional responsibility towards agents c and a, and is liable towards c. Agent e is also liable towards agents c and a. Finally, agent f is liable towards e and has functional responsibility towards e, c and a. Note that if the obligation is fulfilled by an agent belonging to the obligation chain, then we consider that all the other agents have fulfilled their obligations [1]. In addition, we consider that only agents having functional responsibility are obliged to perform the obligation. Otherwise, the obligation is inactivated, i.e. the obligations of agents c and e. After identifying agents that are liable towards b, namely a and e, sanctions are derived. According to the sanction defined by norm x and according to the liability level (i.e. blameworthiness) and kind (i.e. penal or civil liability), a and e

[1] We give here a basic example to illustrate our approach. Real life situations will be given in the next section 4 to help to understand the issues of our work.

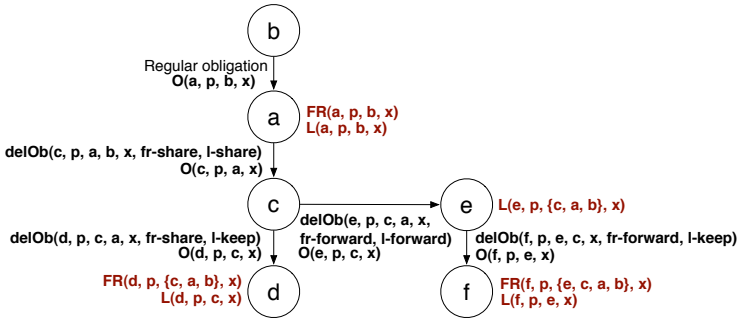


Fig. 1. The obligation chain

will be (or not) sanctioned, asked to repair a damage or a loss. "Local" sanctions can also be defined by norm x for agents who did not fulfill the delegation contract. Thus, agent d, which is "locally" liable to c, can be sanctioned by c if the obligation is violated. Obviously, this liability is inactivated if there is a hierarchical authority relation between the obligatee and the obligator (i.e. $relation(\texttt{bossOf}, d, c)$). In the same way for agents e and f.

4 Related Work and Discussion

In the literature, some works [13,18,6] have studied the issue of the delegation of obligations, such as the share, the transfer or the split of obligations, others [5,17,20,12] have focused on the definition of responsibilities, such as direct, causal or task-based responsibility, and in [7] authors have addressed the issue of accountability within delegation protocols. But, none of them has explored the delegation of responsibility as we have done in this paper. These works have only considered the basic levels of delegation, namely the delegation of the obligation without responsibility or the transfer of the obligation together with the responsibility. In our work, we have proposed a distinction between functional responsibility and liability, in order to give agents the means to delegate their obligations according to their requirements and abilities. Moreover, this distinction allows us to identify, in the case of delegation, agents that are responsible for the violation, agents that are liable (or indirectly liable) for this violation and finally agents who are to be sanctioned.

Even though the word delegation has been used (and defined) in a technical sense in this paper, our notion of delegation can be applied in real life situations in which a proper distinction between functional, causal and legal responsibilities could help clarifying the issues and drawing appropriate conclusions. As an illustration, existing privacy protection regulations (such as the European Directive) impose strong obligations on any entity which collects personal data (the "data controllers"). In particular, data controllers are responsible for the

security of the data and must ensure that the data subject can effectively exercise their rights, for example their rights to get access to their data or to have them corrected in case of error, or deleted if they are no longer necessary for the purpose. If the data controller subcontracts some or all the treatment of the personal data, certain responsibilities are transferred when others are shared or kept by the data controller. For example the functional responsibility to ensure the security of the data is shared (the data controller must implement appropriate security measures for the collection of the data and their transfer to the subcontractor and the subcontractor must ensure the protection of the data storage and access) but the legal responsibility for security (w.r.t. the data subject) may remain with the data controller, so that the data subject could potentially sue the data controller for security breaches actually due to the subcontractor. On the other hand, the functional responsibility on the exercise of the rights of the subject may be completely transferred (if the data controller does not store any data by himself) and both the data controller and the subcontractor must address any request from the subjects concerning their personal data, thus sharing legal responsibility.

Another illustration of the generality of our framework is the application to software contracts and responsibilities for defective software. As stated in section 2, different notions of causal responsibility can be defined, which may correspond to different levels of severity. The notion of causality has also been studied for a long time in computer science, but it is usually seen essentially as a temporal property. In [9], we have defined several variants of logical causality allowing us to express the fact that an event e_2 (e.g. a damage) would not have occurred if another event e_1 had not occurred ("necessary causality") or the fact that e_2 could not be avoided as soon as e_1 had occurred ("sufficient causality"). These causality properties are expressed in terms of execution traces of the software components. We have shown that they are decidable and proposed trace analysis procedures to establish them. These notions of causality are examples of causal responsibilities relations CR which can be used to apply the framework presented here to software liability. This would make it possible to formalize legal aspects of the liability framework proposed in [11] and to distinguish, for example, the technical commitments of a subcontractor (e.g. providing a software component with a given functionality) and the cases of misbehaviour giving rise to a legal liability on his part (e.g. if the output of the component exceeds a given threshold, which might put the system or its environment at risk). An interesting avenue for further work to this respect is the introduction of group liability allowing us to make a distinction between "joint liability" (when each party is considered fully responsible for the obligation) and "several liability" (when the parties are responsible for their respective part of the obligation).

Acknowledgments. This research has been supported by the ANR 07 SESUR FLUOR project.

References

1. Ben-Ghorbel-Talbi, M., Cuppens, F., Cuppens-Boulahia, N.: An extended role-based access control model for delegating obligations. In: Fischer-Hübner, S., Lambrinoudakis, C., Pernul, G. (eds.) TrustBus 2009. LNCS, vol. 5695, pp. 127–137. Springer, Heidelberg (2009)
2. Ben-Ghorbel-Talbi, M., Cuppens, F., Cuppens-Boulahia, N.: Negotiating and delegating obligations. In: International Conference on Management of Emergent Digital Eco-Systems (MEDES) (2010)
3. Bettini, C., Jajodia, S., Wang, X.S., Wijesekera, D.: Provisions and obligations in policy rule management. Network and Systems Management 11(3) (2003)
4. Cholvy, L., Cuppens, F., Saurel, C.: Towards a logical formalization of responsibility. In: 6th International Conference on Artificial Intelligence and Law. ACM Press, Australia (1997)
5. Cholvy, L., Garion, C., Saurel, C.: Ability in a multi-agent context: A model in the situation calculus. In: Toni, F., Torroni, P. (eds.) CLIMA 2005. LNCS (LNAI), vol. 3900, pp. 23–36. Springer, Heidelberg (2006)
6. Cole, J., Derrick, J., Milosevic, Z., Raymond, K.: Author obliged to submit paper before 4 july: Policies in an enterprise specification. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, p. 1. Springer, Heidelberg (2001)
7. Crispo, B., Ruffo, G.: Reasoning about Accountability within Delegation. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 2001. LNCS, vol. 2229, p. 251. Springer, Heidelberg (2001)
8. Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High level conflict management strategies in advanced access control models. Electronic Notes in Theoretical Computer Science, vol. 186 (2007)
9. Gössler, G., Le Métayer, D., Raclet, J.-B.: Causality analysis in contract violation. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G., Roşu, G., Sokolsky, O., Tillmann, N. (eds.) RV 2010. LNCS, vol. 6418, pp. 270–284. Springer, Heidelberg (2010)
10. Irwin, K., Yu, T., Winsborough, W.H.: Assigning responsibility for failed obligations. In: IFIP Trust Management Conference (2008)
11. Le Métayer, D., Maarek, M., Mazza, E., Potet, M.L., Frénot, S., Viet Triem Tong, V., Craipeau, N., Hardouin, R.: Liability in software engineering - overview of the lise approach and illustration on a case study. In: 3rd International Conference on Software Engineering (2010)
12. Mastop, R.: Characterising responsibility in organisational structures: The problem of many hands. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 274–287. Springer, Heidelberg (2010)
13. Pacheco, O., Santos, F.: Delegation in a role-based organization. In: Lomuscio, A., Nute, D. (eds.) DEON 2004. LNCS (LNAI), vol. 3065, pp. 209–227. Springer, Heidelberg (2004)
14. Park, J., Sandhu, R.: The UCON$_{ABC}$ Usage Control Model. ACM Transactions on Information and System Security 7(1), 128–174 (2004)
15. Pretschner, A., Hilty, M., Basin, D.: Distributed usage control. Communications of the ACM 49(9), 39–44 (2006)
16. Pörn, I.: Action theory and social science: Some formal models. Synthese Library 120 (1977)

17. Royakkers, L., Grossi, D., Dignum, F.: Responsibilities in organizations. In: Computer Supported Activity Coordination (2006)
18. Schaad, A., Moffett, J.D.: Delegation of obligations. In: Policies for Distributed Systems and Networks, USA (2002)
19. Sergot, M.: Norms, action and agency in multi-agent systems. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 2–2. Springer, Heidelberg (2010)
20. Strens, R., Dobson, J.: How responsibility modelling leads to security requirements. In: Workshop on New Security Paradigms, United States (1993)
21. von Wright, G.H.: Deontic Logic. Mind 60, 1–15 (1951)

Distributed Security Policy Conformance

Mirko Montanari, Ellick Chan, Kevin Larson,
Wucherl Yoo, and Roy H. Campbell

Department of Computer Science
University of Illinois at Urbana-Champaign
{mmontan2,emchan,klarson5,wyoo5,rhc}@illinois.edu

Abstract. Security policy conformance is a crucial issue in large-scale critical cyber-infrastructure. The complexity of these systems, insider attacks, and the possible speed of an attack on a system necessitate an automated approach to assure a basic level of protection.

This paper presents Odessa, a resilient system for monitoring and validating compliance of networked systems to complex policies. To manage the scale of infrastructure systems and to avoid single points of failure or attack, Odessa distributes policy validation across many network nodes. Partial delegation enables the validation of component policies and of liveness at the edge nodes of the network using redundancy to increase security. Redundant distributed servers aggregate data to validate more complex policies. Our practical implementation of Odessa resists Byzantine failure of monitoring using an architecture that significantly increases scalability and attack resistance.

1 Introduction

Security management and policy compliance are critical issues in modern infrastructure systems. Regulatory and security organizations introduce policies and best practices to raise the minimal level of security required for power grid systems, government systems, and airport systems. We have studied industrial security policies [11, 12] that have complex challenging compliance and auditing concerns at the network level at the scale of the system concerned. Manual attempts to audit these systems are tedious, error prone, and potentially vulnerable to insider attacks or credential theft. Therefore a more principled solution to this problem is required.

The formalization of security policies and the use of hardened automated systems that validate compliance can improve the quality and efficiency of this auditing process. Although previous approaches analyzed the representation of these policies [1] and described centralized systems for collecting network information and analyzing it [6, 16], neither has adequately addressed the issue of scaling to networks of thousands of nodes or of resilience to attacks.

To address these issues, we have implemented and evaluated our policy compliance monitoring system Odessa. Our approach addresses the scaling problem by decomposing policies and distributing the validation process. Each of the

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 210–222, 2011.

complex rules that define the compliant and non-compliant states of the system is decomposed into local components and an aggregate component. We securely delegate the validation of local components to secure agents installed on hosts. These agents are able to reliably monitor the state of the system using virtual machine introspection. Using this information, we partition the validation of aggregate components across several distributed servers. Resilience toward attacks aimed at compromising the validation process uses Byzantine failure resistant, redundant information acquisition employing multiple agents and independent critical policy validation in multiple server style monitors.

The contributions of this paper include:

1. An algorithm for determining which portion of each policy can be validated on devices.
2. A resilient tree-based architecture that distributes to multiple servers the validation of the aggregate components of the policies and that delegates to several hosts the load of monitoring for the liveness of each device.
3. An evaluation of the scalability of our solution.

The rest of the paper is structured as follows. Section 2 describes related work in the area. Section 3 defines policy compliance and presents several examples of policies. Section 4 describes the Odessa architecture. Section 5 presents our algorithm for distributing policy evaluation. Section 6 describes our experimental evaluation. Finally, Section 7 summarizes our contributions and results.

2 Related Work

Several agent-based systems have been introduced for monitoring the security configurations of systems. NetQuery [16] and the DMTF Web Based Enterprise Management (WBEM) framework[1] provide a unified view of the configuration of a system and create notifications in case of changes in the state. However, none of these approaches provide automatic methods for distributing the evaluation of policies or decentralized mechanisms for detecting the failure of hosts.

Other non-agent based systems have been proposed for performing specific security assessments. Network scanners and security assessment tools such as TVA [6], or MulVAL [13] acquire information about the configuration of the system by using port scans or direct access to hosts. These systems have several limitations. First, changes to host configurations are detected with considerable delay because of the polling approach. Second, their architecture is centralized: the evaluation of policy compliance is performed in a central host. For very large networks, this can become both a bottleneck and a vulnerability as a single supervisory node audits, monitors, and checks remote operations that may impact integrity. ConfigAssure [10] takes a top-down approach and synthesizes network configurations from high-level specifications. However, the top-down approach is

[1] http://www.dmtf.org

not always applicable, as the organizational network is often managed by different divisions and it is not always possible to centralize the control into a single entity. Additionally, this paper focuses on policy-compliance validation. Previous work [7] discusses hardening techniques.

3 Policy Compliance

Policy compliance is a basic security and regulatory requirement for infrastructure systems. Although policy compliance cannot guarantee security, it still offers a minimal level of assurance against attacks that would be avoidable had proper security measures been taken. These policies can be specified as constraints created from regulatory requirements, or from the formalization of organization-specific security requirements. Policies are often posed as high-level specifications, and they are implemented in the system using a set of processes or rules that specify constraints on the states of the system. We focus on a set of rules that are in place to protect the industrial infrastructure against a wide-range of known types of attacks. For example, NIST specifies a set of best practices in their Security Content Automation Protocol (SCAP) [12] for regulating the configurations of government systems, and the North American Electric Reliability Corporation (NERC) provides policies about the configuration of power grid systems. For example, a policy might require all remote access points to the system to be monitored and logged at all times, or that all critical systems need to be placed within an enclosed electronic security perimeter. Changes in the configuration of the system or failures could create violations to such security policies and open it to potential attacks.

Many of these policies can be translated into rules and formalized in a logic language [1]. Odessa detects violations of these rules by representing configuration information and state information using Datalog statements [13]. Using Datalog, configurations are expressed as sets of ground statements (i.e., statements without variables). Without loss of generality, we represent configurations using the Resource Description Framework (RDF) language[2] [8]. Following RDF convention, statements are represented as subject-predicate-object triples (s, p, o). A set of statements connected with conjunctions is a knowledge base (KB). For example, we can represent the fact that a server *time.gov* provides the service *ntp* using the following KB: $(time.gov, istype, server)$, $(ntp, istype, service)$, $(time.gov, provides, ntp)$. Statements indicate conditions on the state of a host, and KBs integrate statements to reason about policy violations.

Datalog rules represent implications defined over the state of the infrastructure. The conditions of these implications are specified by a conjunction of *statement patterns*, i.e., statements that have variables as subject or object. Statement patterns are matched (i.e., *unified*) against the statements representing the state, and if the condition is true a new statement is added to the KB. Uppercase characters indicate variables and lowercase characters represent

[2] http://www.w3.org/TR/rdf-concepts/

resources. For example, we can consider a simple rule which specifies that critical servers should not run applications with known vulnerabilities without an exception. By acquiring information about the running program on each machine, annotations about the importance of each server, and information about vulnerabilities, we represent this rule by specifying that we have a violation if a critical server provides a vulnerable service as following: $(S, istype, server)$, $(A, istype, service)$, $(S, provides, A)$, $(S, criticality, high)$, $(A, hasvuln, V)$, $\neg (S, hasexception, E) \rightarrow (r_1, violation, V)$. The last statement, called the *consequence* of the rule, is specified by a statement pattern which have variables that appear in the condition (*body*) of the rule. The consequence can represent a violation as in our example, or it can represent a statement which can be used by other rules for validating compliance.

4 The ODESSA System

The objective of Odessa is to check if the state of the infrastructure as a whole is compliant to all policies defined by the organization and represented as Datalog rules. The architecture of the system was designed to distribute the evaluation of these rules in a scalable manner. To achieve this, Odessa uses a distributed evaluation protocol which includes the following elements: a set of *monitoring agents* that monitor the configuration of hosts and validate local portions of the organizations' policies; a set of distributed *verifiers* that validate global rules, and a set of *predicate groups* which are distributed index structures that provide a scalable support for the communication between verifiers and monitoring agents.

Agents represent the state of hosts using Datalog statements and share statements relevant to global policies with verifiers. Distributed verifiers integrate information across multiple hosts and perform the validation of policies with configuration information gathered from multiple machines. For a subset of policies critical to the security of the infrastructure, we require configuration information to be acquired independently from multiple agents and we replicate the policy validation on several verifiers which use Byzantine fault tolerance [5] to reach an agreement. By virtue of using FreePastry[3], our system inherits secured SSL communications and heartbeat-based liveness measurement. The architecture of Odessa is depicted in Figure 1.

Monitoring Agents. Monitoring agents run in a secure environment on each host. We developed agents running in the `Dom0` VM of a Xen system [2]. Virtual machines running on the same physical hosts are monitored using VM introspection [14]. Traditional hosts are monitored remotely by one of the agents using standard protocols. TPM can be used with SSL to provide a root of trust and validate the authenticity of the agent. The set of policy violations that our system detects depends on the type of information that monitoring agents can acquire from the systems.

[3] http://www.freepastry.org

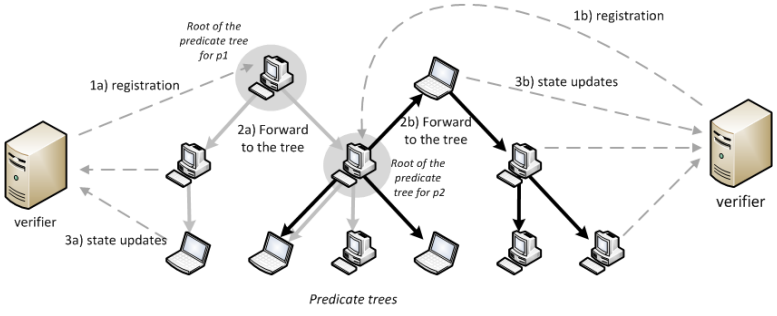


Fig. 1. Architecture of Odessa. End-hosts are organized in predicate groups. Verifiers register to the root of the group.

Verifiers. Verifiers are hosts that securely collect information shared by monitoring agents to validate rules. Each verifier manages a subset of the rules of the organization. The set of rules is partitioned across verifiers to minimize the overlap between the information required by each verifier. Critical rules are analyzed redundantly by multiple verifiers.

Predicate Group. To link monitoring agents and verifiers, we use predicate groups. Each predicate group connects monitoring agents that share a specific type of configuration statements and, hence, participate in the evaluation of the same rules. These groups distribute the processes of distributing information about new verifiers, integrating new hosts, and monitoring their liveness. Monitoring liveness is required because the state of failed hosts needs to be promptly removed from the state of the verifiers to detect correctly policy violations.

A predicate group is formed for every predicate p in the system. Membership in the group is distributed to several hosts by organizing agents into trees: each host maintains knowledge about a few other agents and monitors their liveness. The processes of constructing and maintaining these trees are inspired by the Scribe dissemination trees [4]. Communications are built on a Pastry Distributed Hash Table (DHT) [15] system, and the agent assigned to the DHT key $H(p)$ is the root of the tree for predicate p. When a new verifier starts monitoring compliance to a rule, it contacts the roots of the trees for the predicates involved in the rule to distribute its registration to all agents in the groups.

Resilience. Odessa has several design features that increase the difficulty of attacks when compared to centralized solutions. Attacks can target monitoring agents to compromise the information used for policy validation. To protect them, we run the monitoring agent in a separated secure environment, and we validate critical policies using redundant information. The separation isolates monitoring agents from possible compromises of hosts. In our implementation, we use VM-based separation but other techniques such as hardware based protections can be used without affecting the rest of the architecture. By running only monitoring agents in a separate VM, we provide a smaller attack surface for agents. For these reasons, we assume that agents behave according to our

protocol. However, while techniques have been developed for introspecting the state of machines without malware mediation [14], clever attackers could employ anti-forensics techniques and potentially conceal malicious processes. As an additional level of protection, we use redundant information acquired from independent sources in the validation of critical policies. For example, network traffic can be used to infer the presence of services on a specific machine, and multiple voltage sensors on a power line provide redundant information about the line's state. By acquiring information from multiple agents, an attacker would need to compromise several agents to thwart the validation process.

Attacks can target verifiers to conceal existing policy violations or to insert fictitious violations. We handle these cases by replicating the verification of the same policy on multiple verifiers. We use Byzantine fault tolerance for the validation of critical policies to reach an agreement even when a small subset of the verifiers is compromised. Attacks targeting predicate groups can compromise the infrastructure for distributing new verifiers registration, or for delaying the detection of failed and of new hosts. Even if agents are separated from the monitored hosts, attackers might still able to perform DoS attacks that affect one or more entire physical hosts. However, the DHT infrastructure automatically reconfigures itself to the new situation and notifies verifiers about failed hosts for triggering rule violations. Even when malicious users target the roots of the predicate groups, the DHT reassigns the failed agent role to other agents. Such attack only delays the registration of new verifiers, and it is easily detectable.

5 Rule Decomposition and Validation

To be scalable, our policy validation process detects policy violations through the coordination of monitoring agents and verifiers. We use our rule decomposition algorithm (RDA) to transform the organization's rules into an equivalent set of rules which takes advantage of information locality. This process allows Odessa to push a partial execution of the rules to each monitoring agent and hence, reduces configuration information transferred between machines.

The intuition behind the algorithm is to use information about the source of configuration statements (i.e., which agents can generate a particular configuration statement) for limiting the places where possible configurations that can trigger the body of a rule can be found. For example, if we are checking for the presence of a particular application on a host h_1, we know that information about running applications is generated only by host h_1. Using this locality rationale, we identify a portion of each rule. The execution of this portion that considers only local statements on each agent is equivalent to an execution that considers all statements in the system. Such a portion is executed independently on each agent, and only the results are shared with the rest of the system.

Our validation process is composed of two phases: decomposition and execution. The decomposition phase uses the RDA algorithm to integrate information about the locality of agents' configuration statements with the rules of the organization. The result of this process is a decomposition of each rule into *local sub-rules* and

aggregate sub-rules. In the execution phase, monitoring agents use local sub-rules to perform partial processing of the rule and use predicate groups to send processed state information. Verifiers use aggregate sub-rules to control the process of aggregating information acquired from multiple agents. A more detailed description of the algorithm can be found in the extended version of this paper [9].

5.1 Decomposition

The decomposition phase takes a rule and information about the statements generated by agents to decompose the rule into local and aggregate sub-rules. This process uses an RDF-graph representation of the rules which is more suitable for our analysis. Each rule is transformed in a *rule graph*, a labeled directed graph describing the explicit relationship between variables, resources, and predicates used in the rule. The graph has a node for each resource or variable and an edge from subject to object for every statement pattern. The statement pattern defined in the rule head is marked as the *head edge*.

Locality. For each agent, we say that a statement pattern is *local* if all its potential matching statements are always found locally, independently from the current state of the system. For identifying the local portion of the rule, we formalize our knowledge about the locality of the statement patterns using a RDF graph we call the *locality graph*. One of the nodes of the graph, called the *anchor*, identifies a specific type of agent as the information source (e.g., Host). Each undirected path starting from the root represents a conjunction of statement patterns: all the statements matching such combination of patterns are found locally on each agent. For example, we can consider a path with two statement patterns $(H, hasnetworkinterface, C)$,$(C, connectedTo, N)$. Statements matching these conditions represent the list of network interfaces and networks at which the host H is connected. For a specific host $H = h_1$, the only statements that can match these conditions are generated by h_1. The locality graph depends on the semantics of the statements that are generated by the agent. Statements used in the validation of critical policies should not be part of the local graph.

Using the locality graph we can identify subgraphs of the rule graph which can be processed locally. For clarity, we consider only one type of anchor, `Host`. We generate a different subgraph for each node of type `Host` in the rule graph. We include in this subgraph all edges and nodes that are connected to it which match edges and nodes in the locality graph. We recursively consider all paths in the locality graph. We call these subgraphs *agent-local* subgraphs. Agent-local subgraphs could have anchors which are not local for an agent. For example, given a locality graph (H, p, A) and a rule $(h_2, p, A) \rightarrow (h_2, violation, A)$, the subgraph is local only for host h_2. Without loss of generality, for every agent we choose one of the agent-local subgraph to be *local*.

Transformation into sub-rules. Once the local subgraph is identified, we generate local and aggregate sub-rules to use for the distributed rule processing. These sub-rules specify the location of the computation for the validation of rules and the structure of the communication between agents and verifiers.

A sub-rule is a pair $< \beta \rightarrow \eta, \mu >$ formed by a rule $\beta \rightarrow \eta$ (β is the body, η the conclusion) and a query μ. For local sub-rules, the rule $\beta \rightarrow \eta$ represents a portion of the original rule, and the query μ identifies the statements generated by local processing which are sent to verifiers. For aggregate sub-rules, the query identifies the information received by the agents, and the rule identifies the remaining portion of the policy validation.

Local sub-rules. For each rule graph we consider its local subgraphs. There are several cases. (i) If the local subgraph covers the entire rule $\beta \rightarrow \eta$, then we create a sub-rule $< \beta \rightarrow \eta, \emptyset >$. In this case, the entire processing of the rule can be local and the agent only needs to raise an alarm for violations. (ii) If only the head η statement of the rule graph is not part of the local subgraph, we create a local sub-rule $< \beta \rightarrow \eta, \eta >$. i.e., we locally compute the entire rule, and we share with the verifiers only the consequences. (iii) If the local subgraph covers only a portion of the rule, then we create several local sub-rules. For each edge π of the rule graph not in the local subgraph, we create a local sub-rule $< \emptyset, \pi >$, and we generate a single local sub-rule $< \beta_l \rightarrow \eta_l, \mu_l >$ for the entire local subgraph as follows.

The body of the rule β_l is constructed by taking all edges in the local subgraph and converting them into a conjunctive query of predicate patterns. Because all edges are part of the local subgraph, all statements that match the body of the rule are found locally. For example, each match of a rule body $(A, p_1, B), (B, p_2, C)$ creates a tuple (A, B, C) containing the values of the variables. These tuples need to be shared with the verifiers to complete the validation of the rule. However, we do not need to share the entire set of variables, but only the variables which are used in statement patterns outside the local subgraph. Their variables are identified by considering the nodes at the boundary of the local subgraph (i.e., nodes which have an edge in the local subgraph and an edge outside it). For example, if only the variables A and C are used in statements outside the local subgraph, we only need to share the tuple (A, C).

This tuple, which represents the information to share, is used as the head η_l of the rule. However, because the size of the tuple can change depending on the number of variables in the body, we represent the head using a variable number of statements which share the same RDF blank node as the subject. We can think of blank nodes as resources with a unique name generated automatically by the system. The object of each statement is one of the variables in the body, and the name of the predicate depends on the rule and on the name of the variable. We call these statements *rulematch* statement patterns. For example, the rulematch statements that we define for the body in the example are $(\_ : k, rm_{r,'A'}, A), (\_ : k, rm_{r,'C'}, C)$. The blank node $\_ : k$ is substituted with the same random string for all statements, r is a unique identifier of the rule and $'A'$ and $'C'$ are strings representing the variable names. By combining body and head of the example, we have the rule $(A, p_1, B), (B, p_2, C) \rightarrow (\_ : k, rm_{r,'A'}, A), (\_ : k, rm_{r,'C'}, C)$. The last piece of the local sub-rule, the query μ_l, selects these rule match statements. An example of rule graph and local sub-rules is shown in Figure 2.

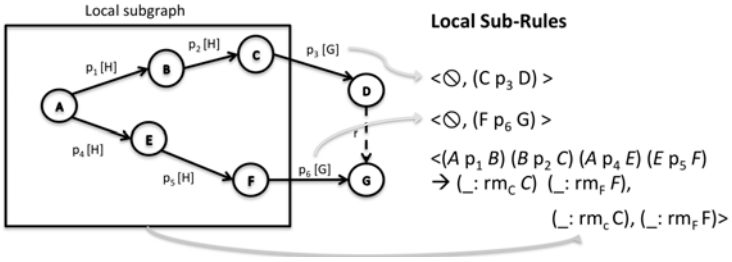


Fig. 2. Example of the conversion of a rulegraph into a set of local sub-rules

Aggregate Sub-Rules. The analysis for the generation of aggregate sub-rules is similar to the generation of local sub-rules. Even if aggregate sub-rules are executed on verifiers, we still use the concept of "locality" as locality for the agents. For edges $\pi = (A, p, B)$ not in the local subgraph we create an aggregate sub-rule with only a query $< \emptyset, \pi >$. This aggregate sub-rule specifies that all statements matching this pattern should be delivered to the verifier. If the rule graph ρ does not have a local subgraph, we add an aggregate sub-rule $< \rho, \emptyset >$ which introduces the rule in the verifier's KB. Hence, for rules with no local subgraphs, the verifiers simply collect all statements matching any of the statement patterns of the rules and perform local reasoning.

For rule graphs with a local subgraph, we need to account for the partial processing performed on the agents. We create an aggregate sub-rule with a rule ρ' where the local subgraph edges have been substituted with rulematch statements, and we create a set of queries that collects such statements.

5.2 Execution

The execution is driven by local sub-rules and aggregate sub-rules. For each aggregate sub-rule $< \rho, \mu >$ the verifier adds the rule ρ to its local KB. Considering $\mu = (A, p, B)$, it sends a message to the root of the predicate group of p, $H(p)$. The message contains the query μ which the root nodes disseminates to all agents in the group. On the agents, for each local sub-rule $< \rho, \mu >$ we add the rule ρ to the local KB and we select all statements matching μ. Assuming $\mu = (A, p, B)$, the agent sends a message toward $H(p)$ to register itself as part of the predicate group. In the DHT, the path toward such a node travels through multiple other monitoring agents. At each step, agents on the path keep track of the subscription and form a new branch of the tree used to maintain the predicate group. When an agent that is already part of the same tree is found, the branch is connected as a child of such agent. The agent receives from the new parent the registered verifiers and sends them its configurations.

For the validation of critical policies, we require verifiers to collect statements from a minimum number of different agents. A Byzantine agreement on the result is reached by broadcasting the result of local validations to other verifiers.

The DHT infrastructure supports the monitoring for liveness. Each monitoring agent is registered to several predicate groups, and the agents periodically send heartbeat messages to their neighbors. When the failure of an agent is detected, a message is sent to the registered verifiers so that all statements that had been generated by the failed agent are removed from the verifiers' state. As each host is registered to several trees, even the failures of all hosts in a branch of the tree are detected by the parent hosts in other trees.

6 Implementation and Evaluation

We implemented the components of the Odessa system using a combination of C and Java. The communication between monitoring agents and verifiers is implemented on the FreePastry system. Inference is performed by using the rule-based inference system Jena [3]. The monitoring agents run in the `Dom0` virtual machine of a Xen installation. They monitor guest VMs by accessing the host state using an extension of XenAccess [14]. A Linux kernel module is installed on guest VMs to provide additional information about the state of the system which is not easily accessible using XenAccess.

We ran the system on a real network and we validated a set of test rules which include (i) checking the presence of specific programs, (ii) checking NFS authorization for access control misconfigurations that give unprivileged users access to restricted files, (iii) and validating that critical machines are protected from external attacks. Our system was able to delegate the validation of rules (i) and (ii) to each host, and it was able to decompose rule (iii) into a local portion and a global portion. The local portion shares statements about the host address and about vulnerable programs running on the system, which are identified using the National Vulnerability Database (NVD)[4]. The global portion integrates this information across the network and using logic attack graphs it computes if a specific host can be compromised by an external attacker. We use our prototype to measure the possible delay in the verification that an attacker can introduce by performing DoS on predicate group roots before a new verifier is registered. We found that the FreePastry implementation already provides a delay limited to an average in the order of tens of seconds. The tradeoff between message frequency and delay in the detection of failures is shown in Figure 3. The parameter p represents the number of communication attempts made before declaring an agent dead. The results are an average of 20 executions.

To measure the scalability characteristics of Odessa, we performed several simulations using random models of large-scale systems. The first experiments focus on the ability of Odessa of distributing rule validations, and the second on the scalability of the system. We use the number of statements shared by monitoring agents as a metric for measuring the distribution of rule validation. We create a synthetic configuration with a structure similar to the configuration data found in real hosts (e.g., process list and associated sockets, network file system data). The configuration is composed of a constant number of statements organized in

[4] NVD: National Vulnerability Database V2.2 http://nvd.nist.gov/

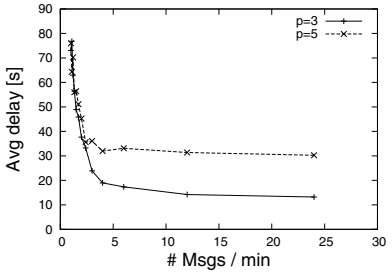


Fig. 3. Delay in the detection of agent failures

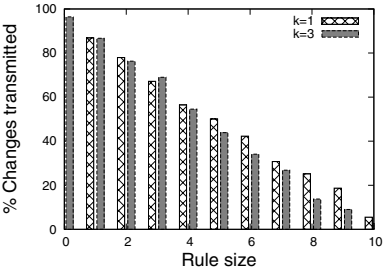


Fig. 4. Agents' statements transferred as consequences of configuration changes

tree structures where the object of the parent statement is the subject of the children. We vary the number of children at each level with a parameter k, and we vary the number of levels. Random statements have constants as objects. We consider a rule body (A_1, p_1, A_2), ..., (A_i, p_i, A_{i+1}), ..., (A_m, p_m, A_{m+1}) and we changed the local sub-rule by varying the index i to represent different types of configurations and to represent the use of critical policies (which decrease the local portion of the rule). We consider a system where agent configurations change periodically by randomly adding or removing statements and we measure the effects of the size of the sub-rule in the number of statements transmitted. We found that the number of statements decreases linearly with the increase of the local portion of the rule, as shown in Figure 4.

The next set of experiments shows that, independent from the advantages of delegating rule processing, the use of predicate groups significantly reduces the load on the verifiers for monitoring the liveness of hosts and, hence, enables an increased scalability of the system. To quantify this gain, we perform simulations to compare our architecture with an agent-based solution which relies on a central server for integrating data. We set the parameters of the two solutions (e.g., frequency of messages used for keep-alive purposes) to obtain an equivalent delay in the detection of failed hosts, and we consider both a static network (`odessa-p50-static`) and a network where a host is added or removed

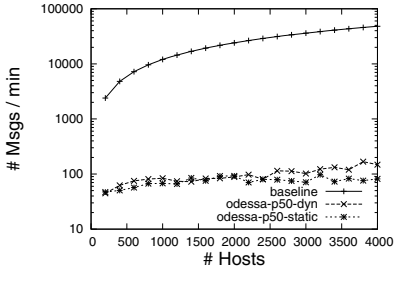


Fig. 5. Maximum number of messages sent and received by any hosts (log y)

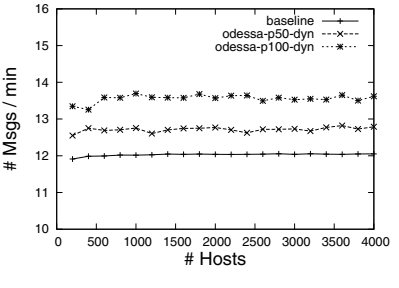


Fig. 6. Average number of messages sent by each single host

from the system so that 20% of the hosts change every hour (`odessa-p50-dyn`). We measure the maximum amount of messages sent and received by each host. We find that our solution reduces by orders of magnitude the maximum load on any single host (shown as in Figure 5) and has a limited effect on the average load of each single host (shown in Figure 6). We also find that the number of predicate groups at which hosts are connected does not significantly affect the average number of messages exchanged. In the figures, `odessa-p100-dyn` represents a network where each host is connected to 100 predicate groups, while in `odessa-p50-dyn` each host is connected to 50 predicate groups.

7 Concluding Remarks

This paper shows that resilient and large-scale policy validation is possible by introducing an architecture and an algorithm for decomposing policies and distributing their validation across multiple machines. We assess that our technique is viable and practical for deployment in large infrastructure systems.

In our future work we are planning to employ reactive agents that can harden the host according to security policies to reduce the time window of vulnerability of the system. Our approach focuses on rules for which violations have a long lifespan. Short-lived false negatives from message reordering pose a limited threat to security because they already have a small time window for attack. However, we are planning to address these issues for a more general monitoring system.

References

1. Anwar, Z., Campbell, R.H.: Automated Assessment of Critical Infrastructures for Compliance to CIP Best Practices. In: Second IFIP WG 11.10 International Conference on Critical Infrastructure Protection. IFIP (2008)
2. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP. ACM, New York (2003)
3. Carroll, J., Reynolds, D., Dickinson, I., Seaborne, A., Dollin, C., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: WWW. ACM, New York (2004)
4. Castro, M., Druschel, P., Kermarrec, A., Rowstron, A.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications (2002)
5. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS) (1982)
6. Jajodia, S., Noel, S., Berry, B.: Topological analysis of network attack vulnerability. In: Managing Cyber Threats: Issues, Approaches and Challenges (2005)
7. Johnson, C., Montanari, M., Campbell, R.H.: Automatic Management of Logging Infrastructur. In: CAE Workshop on Insider Threat. CAE (2010)
8. Montanari, M., Campbell, R.H.: Multi-Aspect Security Configuration Assessment. In: SafeConfig Workshop. ACM, New York (2009)

9. Montanari, M., Chan, E., Larson, K., Yoo, W., Campbell, R.H.: Distributed Security Policy Conformance. Technical Report, University of Illinois (February 2011)
10. Narain, S., Levin, G., Malik, S., Kaul, V.: Declarative Infrastructure Configuration Synthesis and Debugging. Journal of Network and Systems Management (2008)
11. North American Electric Reliability Corporation, Critical Infrastructure Protection Standard, CIP-001 to CIP-009 (2010)
12. NIST. SP800-126: The Technical Specification for the Security Content Automation Protocol (SCAP) (2009)
13. Ou, X., Boyer, W., McQueen, M.: A scalable approach to attack graph generation. In: CCS. ACM, New York (2006)
14. Payne, B.D., Carbone, M., Lee, W.: Secure and Flexible Monitoring of Virtual Machines. In: ACSAC. IEEE, Los Alamitos (2007)
15. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Liu, H. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001)
16. Shieh, A., Kennedy, O., Sirer, E., Schneider, F.: NetQuery: A General-Purpose Channel for Reasoning about Network Properties. In: OSDI. USENIX (2008)

Scalable Privacy-Preserving Data Mining with Asynchronously Partitioned Datasets

Hiroaki Kikuchi[1], Daisuke Kagawa[1], Anirban Basu[1], Kazuhiko Ishii[2], Masayuki Terada[2], and Sadayuki Hongo[3]

[1] Graduate School of Engineering, Tokai University,
1117, Kitakaname, Hiratsuka, Kanagawa, 259-1292, Japan
kikn@tokai.ac.jp, {nagomin03,abasu}@cs.dm.u-tokai.ac.jp
[2] NTT DoCoMo Inc.
3-5 Hikarinooka, Yokosuka-shi, Kanagawa, 239-8536, Japan
{ishiikaz,teradama}@nttdocomo.co.jp
[3] Department of Frontier Information Engineering,
Faculty of Advanced Engineering, Hokkaido Institute of Technology 7-15 Maeda,
Teine-ku, Sapporo, Hokkaido, 006-8585, Japan
hongo@hit.ac.jp

Abstract. In the Naïve Bayes classification problem using a vertically partitioned dataset, the conventional scheme to preserve privacy of each partition uses a secure scalar product and is based on the assumption that the data is synchronised amongst common unique identities. In this paper, we attempt to discard this assumption in order to develop a more efficient and secure scheme to perform classification with minimal disclosure of private data. Our proposed scheme is based on the work by Vaidya and Clifton[1], which uses commutative encryption to perform secure set intersection so that the parties with access to the individual partitions have no knowledge of the intersection. The evaluations presented in this paper are based on experimental results, which show that our proposed protocol scales well with large sparse datasets.

1 Introduction

Privacy-preserving data mining aims to allow computation of useful aggregate statistics over the entire dataset without compromising the privacy of individual data. The parties collaborating to obtain aggregate results may not fully trust each other, such as a Sybil attack[2] resistant recommendation system [3] or the Naïve Bayes classifier [1]. Such parties may also be competitors in the same field, for example companies which may have privacy policies restricting access to each other's customer datasets.

Vertically partitioned data is an important data distribution model often found in real life. For example, Table 1 illustrates two datasets partitioned vertically where attributes A_1 and A_2 are owned by Alice (A); and Bob (B)[1] owns the attribute A_3 and a target class C, which indicates whether or not to play tennis on

[1] From this point forward, we use Alice, A, and party A interchangeably; and the same for Bob, B or party B.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 223–234, 2011.

the day. Alice and Bob separately collect the different features, e.g. temperature, humidity, etc. for each day. Collaboratively performing Naïve Bayes classification allows them to accurately predict the decision to play or not, i.e. predict C given A_1, A_2 and A_3, although they can not share each other's datasets.

Syncronous and Asynchronous Partitions: Vaidya and Clifton presented, in [1], a secure protocol for Naïve Bayes classification for vertically partitioned datasets without revealing the individual partitions. Their protocol combines homomorphic public-key encryption algorithm to compute scalar product of two vectors, with the secure function evaluation [4] for comparison of class $c \in C$ in terms of conditional attributes, i.e. $Pr(C = c|a_1, a_3)$.

Their protocol assumes that the input vectors are of the same dimensions. Hence, the partitioned datasets are synchronous with the days (in our example in Table 1) when the attributes are observed. However, datasets may not always be synchronous. For example, the dataset in Table 2 is vertically but asynchronously partitioned, where attributes are stored with common IDs. This type of asynchronous partitions are of frequent occurrences in our daily lives. Examples include some content service providers with common user IDs, while hospitals and pharmacies may share some common patient identities.

Before delving further, we define few terms that we use in this paper:

asynchronous partitions are vertical partitions of a dataset which are more generalised cases of synchronous partitions. Asynchronous partitions do not necessarily exhibit a coherent sequence of data between the partitions, for example, by having missing and duplicate instances, or not being indexed by the same identity column.

index set is a set, denoted by ID, of values for identities of all instances in a dataset or its partition.

Table 1. Synchronously (vertically) partitioned dataset

	Alice		Bob	
day	A_1	A_2	A_3	C
1	sunny	hot	high	no
2	sunny	hot	low	yes
3	rainy	hot	high	yes
4	rainy	cool	low	yes

Table 2. Asynchronously partitioned dataset

	Alice			Bob	
ID	A_1	A_2	ID	A_3	C
1	sunny	hot	1	high	no
2	sunny	hot	-	-	-
3	rain	hot	3	low	yes
4	rain	cool	4	low	yes
-	-	-	5	high	yes
-	-	-	3	high	yes

The simplest solution to the problem of asynchronously partitioned datasets is to sort instances by IDs so as to make the two datasets consistent by IDs, and then perform secure scalar product protocols on the vectors. However, it is not so easy. Attributes may be missing for certain IDs, e.g. for $id = 2$ in Bob's partition. One ID may have multiple instances, e.g. the 3rd and the 6th instances conflict for the common $id = 3$ in Bob's partition. The most significant issue in asynchronous partitions is scalability. The conventional vector-based approaches are not efficient for datasets in which most instances are empty, i.e. sparse datasets and this leads us to what is often called the *sparsity problem*, e.g. [5].

missing values. Datasets may consist of missing values for some IDs, which contributes to low density of data. For example, the density is only 0.03 in "EachMovie" dataset[2] [6]!

duplicate assignment. Datasets may assign the same ID to distinct instances. The arbitrary assignment of IDs can make datasets inconsistent.

scalability. Vector-based approach requires processing for each element of the input vector even if most elements are empty. Such schemes do not scale well as the computational costs increase dramatically with increases in dataset size in sparse datasets.

Our goal and approaches: The goal of this paper is to construct a privacy-preserving protocol for applying the Naïve Bayes classifier to vertically and asynchronously partitioned datasets, which deals with the issues of (1) missing values, (2) duplicate assignments, and (3) scalablity.

In order to address these issues in asynchronous partitions, we introduce the secure set intersection scheme presented by Agrawal et al. in [7], which uses commutative public-key encryption. The particular advantage of the scheme is that it works only on non-zero elements and is, therefore, appropriate for sparse datasets.

The contributions of this paper are: (1) first work of its kind, to our knowledge, to consider asynchronously partitioned datasets; (2) a secure privacy preserving scheme which scales well with the size of data and works efficiently with sparse datasets; (3) a performance based evaluation of an experimental implementation of the proposed scheme; and (4) an analytical evaluation of the scheme in terms of the how much information is revealed.

Organisation: This paper is organised as follows. In Section 2, we review some of the fundamental concepts and the existing work in privacy-preserving data mining. In section 3, we present our proposed scheme. In Section 4, we evaluate our scheme based on an experimental implementation.

2 Building Blocks

2.1 Naïve Bayes Classifier

Naïve Bayes is a widely used classifier based on the Bayes theorem, where the class with the highest likelihood is chosen. Since the algorithm is simple but efficient to implement, it is widely used for many purposes including email spam filtering, prediction of credit scoring, and so on.

[2] EachMovie dataset in the Grouplens project: http://www.grouplens.org/node/76

2.2 Secure Scalar Product Based Scheme

Vaidya and Clifton proposed a privacy-preserving scheme for the Naïve Bayes classifier in [1]. The method allows two parties, each having access to only one partition of a vertically partitioned dataset to predict the most likely target class for any given instance without revealing data from each other's partitions.

Algorithm 1. Secure Scalar Product

Input: Alice has n-dimentional vector $\boldsymbol{x} = (x_1, \ldots, x_n)$. Bob has n-dimentional $\boldsymbol{y} = (y_1, \ldots, y_n)$.
Output: Alice has s_A and Bob has s_B such that $s_A + s_B = \boldsymbol{x} \cdot \boldsymbol{y}$.

1. Alice generates a homomorphic public-key pair and sends public key to Bob.
2. Alice sends to Bob n ciphertexts $E(x_1), \ldots, E(x_n)$.
3. Bob chooses s_B at random, computes
$$c = E(x_1)^{y_1} \cdots E(x_n)^{y_n} / E(s_B)$$
and send c to Alice.
4. Alice decrypts c to get $s_A = D(c) = x_1 y_1 + \cdots + x_n y_n - s_B$.

The drawback of the protocol is the strong assumption of a *synchronous partition*, i.e. (1) vectors $\boldsymbol{a}$ and $\boldsymbol{c}$ have the same dimension, (2) elements correspond to each other for two vectors.

The secure scalar product based methods, hence, can not simply be applied to asynchronously partitioned datasets such as Table 2.

3 Proposed Scheme

3.1 Idea

In order to perform Naïve Bayes in a scalable way, we introduce a secure set intersection protocol, which allows Alice and Bob with subsets X and Y, respectively, to compute $X \cap Y$ without revealing X or Y. Intersection is an useful primitive for many data mining algorithms and hence has been studied so far in [8,9]. The scheme presented in [8] uses oblivious polynomial evaluation that suffers from the linear relation between computational cost and the order of the polynomial. It is, therefore, not appropriate for our purpose. For our study, we focus on the scheme presented by Agrawal, et. al. in [7], which uses commutative public-key encryption, which is performed only for active (i.e. not missing) elements and therefore is more appropriate for sparse datasets.

The aforementioned intersection protocol, however, reveals intermediate results to get the final prediction for the class variable, because one party must learn how many elements belong to both Alice and Bob in order to proceed with the protocol. On the other hand, the existing secure scalar product preserves the secrecy about the size of intersection $|X \cap Y|$ through an additional random number (s_B at Step 3 in Algorithm 1). The revealed information is critical to privacy preservation. Therefore, we propose a new secure protocol based on [7] in order to improve both privacy and scalability for privacy-preserving data mining.

3.2 Secure Set Intersection Protocol

Agrawal, et. al. proposed, in [7], a secure intersection protocol using a public-key encryption algorithm that is commutative, i.e. $f(g(x)) = g(f(x))$ and proved its security under assumption of semi-honest model and random-oracle model. For concrete discussion, we illustrate the scheme in Algorithm 2 using a power function $f_e(x) = x^e \bmod p$ defined under Decisional Diffie-Hellman hypothesis as commutative encryption[3].

Algorithm 2. Secure Intersection Protocol

Input: Alice has subset $X = \{x_1, \ldots, x_{n_A}\}$, Bob has subset $Y = \{y_1, \ldots, y_{n_B}\}$.
Output: Intersection $|X \cap Y|$.

Let Z_q be a multiplicative group with prime order q and H be a secure hash function that maps into range G.

1. Alice chooses random $u \in Z_q$ and send to Bob $H(x_1)^u, \ldots, H(x_{n_A})^u$ in random order.
2. Bob chooses random $v \in Z_q$ and send to Alice $H(y_1)^v, \ldots, H(y_{n_B})^v$ and $(H(x_1)^u)^v, \ldots, (H(x_{n_A})^u)^v$ as well.
3. Alice computes $(H(y_1)^v)^u, \ldots, (H(y_{n_B})^v)^u$ and selects pairs (x_j, y_i) such that $H(y_i)^{vu} = H(x_j)^{uv}$; the number of pairs being the size of intersection $= |X \cap Y|$.

Algorithm 2 with an input of set of n elements requires n hash value evaluation, $2n$ modular exponentiations for each party[4], and n-element set comparison, which runs in $n \log n$ time with any appropriate algorithm. So, total complexity is $O(n) + O(n \log n) = O(n \log n)$, but the most significant cost is that for modular exponentiation. Supposing t_e be a processing time for exponentiations, the cost of $2n$ exponentiations is $2nt_e$.

While the polynomial interpolation based algorithm in [8], known as popular intersection protocol, requires $O(n \log \log n)$ modular exponentiations for oblivious polynomial evaluation, we will show, later in this paper, that the computational cost is considerably large and hence the commutative encryption is proper for large-scale data mining.

3.3 Proposed Protocol: Distorted Intersection

The goal of our protocol is to compute a conditional probability of $c \in C$ given $a \in A_j$, $Pr(c|a)$, where party A has the attribute A_j and B has the target class C. We denote as index sets, X and Y, defined over the ranges of A_j and C, as

$$X_{a,A_j} = \{id \in ID | A_j(id) = a\}, \; Y_c = \{id \in ID | C(id) = c\}.$$

[3] For easy understanding, we describe a simplified protocol where only Alice learns the results.

[4] $n = max(n_A, n_B) + \epsilon$ where ϵ is a positive integral constant; thus n is bigger than both n_A and n_B.

For instance, the datasets in Table 2 define the corresponding index sets for $a = sunny$ and $c = yes$ as $X_{sunny,A_1} = \{1, 2\}$ and $Y_{yes} = \{3, 4, 5\}$ respectively.

In order to hide the size of intersection from Alice (A), Bob (B) wishes to add random noise to his secret input. However, B does not know which elements belong to the intersection prior to the execution of the protocol. Hence, he makes his own input distorted by discarding some elements with random probability $p = s_B/n_B$ so that A cannot learn the exact size of the intersection without knowledge of the random probability distribution.

With the randomisation step, the resulting size of the intersection is skewed with p as $s_A = |X \cap Y| s_B/n_B$, which is known to A who does not know p; while, B knows p but does not know s_A. Therefore, both parties participate in Yao's secure multi-party protocol to compute the multiplication

$$s_A \cdot \frac{n_B}{s_B} = |X \cap Y|,$$

which gives the conditional probability

$$Pr(X|Y) = \frac{|X \cap Y|}{|C|}.$$

Finally, the prediction of target class for a given instance, c_{NB}, is obtained from Naïve Bayes classifier. Yao's protocol allows them to compare several candidates of the class without revealing any partial intermediate information.

Our proposed protocol is described in Algorithm 3.

4 Evaluation

4.1 Performance evaluation

In order to evaluate performance improvement of the proposed scheme in comparison with others, we implement the following schemes for the secure Naïve Bayes classifier:

1. Scalar product based scheme, Vaidya and Clifton [1], which requires a homomorphic encryption and a secure function evaluation of comparison of additively shared value (SFE_1),
2. Set intersection schemes, proposed by Freedman, Nissim and Pinkas [8], requiring a secure polynomial evaluation, and
3. Commutative encryption scheme, proposed in this paper, which requires a homomorphic encryption and a secure function evaluation of comparison of multiplicatively shared value (SFE_2).

Test implementation. Our trial experimental system is implemented using Java (SDK 1.6.0) running on Intel Core2 Duo CPU 2.53 GHz, 2GB, Windows 7 (32 bit). We use the Paillier encryption with $|n^2| = 2048$ bit modulus for additive homomorphic property, with a proprietary public key format. Our implementation has the average processing time of our trial implementation for encryption, decryption and modular exponentiation, denoted by $t_E = 1.1$, $t_D = 1.6$ and $t_P = 0.15$, respectively. Note that the cost of decryption is higher than that of encryption because of property of Paillier encryption [10].

Algorithm 3. Distorted Intersection

Input: Alice has subset $X = \{x_1, \ldots, x_{n_A}\}$, Bob has subset $Y = \{y_1, \ldots, y_{n_B}\}$.
Output: shares of intersection, such that $s_A \cdot s_B = |X \cap Y|$.

1. Alice chooses random $u \in Z_q$, computes $H(x_1)^u, \ldots, H(x_{n_A})^u$ and send to Bob in random order. Alternatively, she can sort these values in numerical order.
2. Bob chooses random $v \in Z_q$, computes $H(x_1)^{uv}, \ldots, H(x_{n_A})^{uv}$ and send to Bob in random order.
3. Bob chooses random $s_B (< n_B)$ and for $i = 1, \ldots, n_B$, compute

$$w_i = \begin{cases} H(y_i)^v & \text{with probability} = s_B/n_B, \\ r_i & \text{otherwise,} \end{cases}$$

   where r_i is randomly chosen from Z_q except $H(y_i)^v$ for every i. Then Bob sends $w_1, \ldots, w_{n_B}$ in random order to Alice.
4. Alice finds pairs x_j, y_i such that $H(y_i)^{vu} = H(x_j)^{uv}$, and where s_A is the number of pairs, i.e. $s_A (= |X \cap Y|(s_B/n_B))$.

Secure Scalar Product. The secure scalar product based scheme requires N encryptions and one decryption plus secure function evaluation of comparison of shared sum (SFE_1), i.e.

$$T_1 = Nt_E + t_D + SFE_1$$

where N is the dimension of the vectors. Note that mostly $N \gg n$, where n is the number of active IDs in the asynchronously partitioned dataset.

Secure Function Evaluation (SFE). We use the generic two-party secure function evaluation evaluation system, Fairplay[11]. Fairplay consists of a compiler of a high level procedural definition language, SFDL, into a one-pass Boolean circuit in a language called SHDL.

With Fairplay, we can perform secure function without revealing inputs. Figure 1 is the source code 'SharedCmp' to securely test $s_{A0} + s_{B0} > s_{A1} + s_{B1}$ where s_{A0}, s_{A1} are owned by Alice and values s_0 and s_1, additively shared as $s_0 = s_{A0} + s_{B0}$ are compared. The bit size is 16.

The example shows that Fairplay allows us to code arbitrary functions easily. However, due to the processing cost, multiplication and division are not provided as primitive operations[11]. We have to code those as programmed functions to perform comparison for multiplicatively shared values as $s_{A0} \cdot s_{B0} > s_{A1} \cdot s_{B1}$. (In our trial implementation, we omit the division since we can replace it by multiplication with some constant).

Table 3 shows the average processing time measured by Alice and Bob for several classes. Both parties have almost the same overhead to jointly evaluate comparison. In this experiment, we use 16-bit integers.

```
program SharedCmp {
  const size = 20;   type int = Int<16>;
  type AliceInput = int[size]; type BobInput = int[size];
  type AliceOutput = int;  type BobOutput = int;
  type Output = struct {AliceOutput alice,   BobOutput bob};
  type Input = struct {AliceInput alice,   BobInput bob};
  function Output output(Input input) {
    if(input.alice[0] + input.bob[0] > input.alice[1] + input.bob[1]){
      output.alice = input.alice[0];
      output.bob = input.bob[0];
    }else{
      output.alice = input.alice[1];
      output.bob = input.bob[1];
    }
  }
}
```

Fig. 1. Fairplay program (in SFDL) 'SharedCmp': shared integer comparison $s_{A0} + s_{B0} > s_{A1} + s_{B1}$

Table 4 gives our estimation of performance for SFE_1 (addition) and SFE_2 (multiplication) based on the experimental measurement. Based on the runtime complexity, the curve fitting polynomial in terms of size of input x and the processing time when $x = 10bits(= 1024)$ are given. Figure 2 illustrates the estimation. We observe that the cost for secure multiplication increases with respect to the input size, and hence the our proposed scheme has a considerable large constant time overhead.

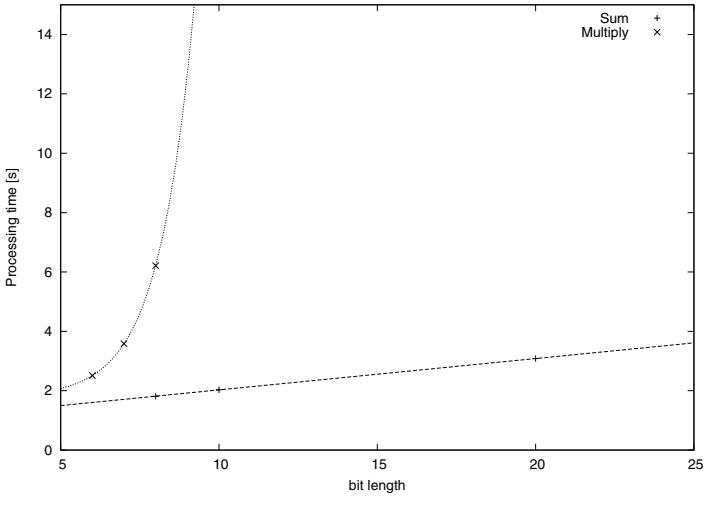


Fig. 2. Processing Time in Yao's SFE (Secure Function Evaluation) for sum and multiplication

Table 3. Processing Time SFE_1 (Shared Comparison)

	Alice	Bob
Mean processing time (sec)	0.80	0.82

Table 4. Processing Time for SFE_1 (addition) and SFE_2 (multiplication) with size of 10 bit (= 1024) interger

circuit	fitting	Time (sec)
SFE_1 (addition)	$0.97 + 0.106x$	2.03
SFE_2 (multiplication)	$1.77 + 0.003e^{0.89x}$	23.76

Scalability of Proposed Scheme. Our proposed Algorithm 3 requires as many encryptions as the number of active users, n, and runs in time

$$T_2 = 2t_P n + t_c n \log n + SFE_2,$$

where t_c is the cost of comparison of n size lists. We may omit the overhead for comparison because $t_c \ll t_e, t_d$. Since $n \ll N$, it runs faster than the secure scalar product based protocol (T_1). However, the constant overhead for secure function evaluation of multiplicatively shared values (SFE_2) is higher than that of additive shared values (SFE_1). The proposed protocol is scalable in terms of the entire size of dataset, N, but suffers the constant overhead of SFE.

Therefore, we conclude that the proposed scheme is efficient only for large sparse datasets such that

$$N^* \geq \frac{SFE_2 - SFE_1 - t_D}{t_E - 2t_P\alpha},$$

where $\alpha = n/N$ assuming n is proportional to the entire size of dataset. We illustrate the scalability of our proposed scheme in Figure 3, where the proposed one is more efficient than the secure scalar product based scheme [1] when N is large. Most of asynchronously partitioned datasets are considered as ones with small fractions of intersection. Consequently, we can say that the proposed scheme improves the performance for large scale sparse datasets.

4.2 Security

In [7], assuming the random oracle model and no hash collisions, and in semi-honest model, there is no polynomial-time algorithm that can distinguish between a random value and $H(x)^u$ given x. This means that Algorithm 2 preserves the privacy of input subsets X and Y. With zero-knowledge proof, the security in the random oracle model can even be extended to a malicious model where parties behave arbitrarily.

Algorithm 1 is also proved as secure even after one party (Alice) learns the result of the protocol, $s_A = x_1y_1 + \cdots + x_ny_n - s_B$, which is randomised with

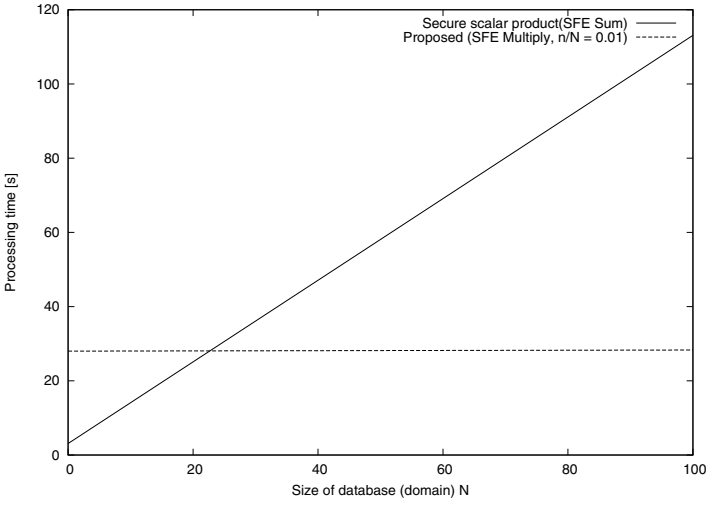


Fig. 3. Scalability in terms of processing time for the size of database (set of indexes) N

s_B chosen uniformly by the other party (Bob). More formally, learning partial result s_A reveals nothing about the distribution of $s_A + s_B$, which is distributed uniformly over group Z_q of order q, that is, the conditional probability of the sum given s_A is identical to apriori probability, i.e. $Pr(s_A + s_B|s_A) = Pr(s_A + s_B) = 1/q$. SFE also preserves the secrecy of shared inputs under the assumptions of semantically secure public key algorithm.

However, the distortion in Algorithm 3 is not uniform. Let z be the size of intersection $|X \cap Y|$, and p be a probability to apply commutative encryption in the algorithm, defined as $p = s_B/n_B$. The conditional probability of the algorithm outputs s_A given z is computed with the binomial distribution as:

$$Pr(s_A|z) = \begin{cases} 0 & \text{if } s_A > z, \\ \binom{z}{s_A} p^{s_A}(1-p)^{z-s_A} & \text{otherwise.} \end{cases} \quad (1)$$

Then, what can Alice guess about z after she learns the output of the algorithm, s_A?

Table 5. Summary of proposed scheme

scheme:	Vaidya & Clifton [1]	Proposed
based on:	Secure Scalar Product [12]	Commutative encryption [7]
input	N-dimension binary vectors	integer subset of size n
computation cost	$T_1 = t_E N + t_D + SFE_1$	$T_3 = t_P n + n \log n + SFE_2$
accuracy	accurate	accurate with probability s_B/n_B
security	$Pr(z|X.Y) = 1/N$	$Pr(z|s_A) > 1/n$

Bayes theorem gives to her an useful hint, i.e. the probability distribution of z as

$$Pr(z|s_A) = \frac{Pr(s_A|z)Pr(z)}{Pr(s_A)} = \frac{Pr(s_A|z)1/(n+1))}{\sum_z Pr(s_A|z)Pr(z)},$$

where we assumes $Pr(z) = 1/(n+1)$.

5 Conclusion

We have proposed a scalable privacy-preserving Naïve Bayes classifier for asynchronously partitioned datasets. Our proposed scheme is based on the work presented in [7] using a public-key encryption algorithm that satisfies commutative property. The performance of our proposed protocol is shown to be better than the scheme based on the secure scalar product [1] when the matrix is sparse, i.e. most entries are missing and the fraction of active data is small, that is $n \ll N$, which frequently happens in asynchronously partitioned datasets. Table 5 gives the summary of the features of our proposed protocol.

References

1. Vaidya, J., Clifton, C.: Privacy Preserving Naïve Bayes Classifier for Vertically Partitioned Data. In: SIAM International Conference on Data Mining, Lake Buena Vista, Florida, pp. 522–526. Society of Industrial and Applied Mathematics, Philadelphia (2004)
2. Douceur, J.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
3. Yu, H., Shi, C., Kaminsky, M., Gibbons, P.B., Xiao, F.: DSybil: Optimal Sybil-resistance for Recommendation Systems. In: 30th IEEE Symposium on Security and Privacy, pp. 283–298. IEEE, Los Alamitos (2009)
4. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE, Los Alamitos (1986)
5. Zhou, J., Luo, T.: A novel approach to solve the sparsity problem in collaborative filtering. In: International Conference on Networking, Sensing and Control (ICNSC), pp. 165–170. IEEE, Los Alamitos (2010)
6. GroupLens: GroupLens Research, http://www.grouplens.org/ (2010)
7. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: The ACM SIGMOD International Conference on Management of Data, pp. 86–97. ACM, New York (2003)
8. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
9. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. Journal of Computer Security 13(4), 593–622 (2005)
10. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
11. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay – a secure two-party computation system. In: The 13th USENIX Conference on Security Symposium, p. 20. USENIX Association (2004)

12. Du, W., Atallah, M.J.: Privacy-preserving cooperative statistical analysis. In: 17th Annual Computer Security Applications Conference, ACSAC, pp. 102–110. IEEE, Los Alamitos (2001)
13. Kikuchi, H., Kizawa, H., Tada, M.: Privacy-Preserving Collaborative Filtering Schemes. In: International Conference on Availability, Reliability and Security, ARES 2009, pp. 911–916. IEEE, Los Alamitos (2009)
14. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: The 10th International Conference on World Wide Web, pp. 285–295. ACM, New York (2001)

Privacy-Enhanced Web-Based Event Scheduling with Majority Agreement

Benjamin Kellermann

Technische Universität Dresden, Faculty of Computer Science,
D-01062 Dresden, Germany
Benjamin.Kellermann@tu-dresden.de

Abstract. Applications which help users to schedule events are becoming more and more important. A drawback of most existing applications is, that the preferences of all participants are revealed to the others. Previously proposed privacy-friendly solutions could only schedule meetings if all participants were available at the same time slot.

We propose a new scheme, which overcomes this limitation, i.e., the meeting can be scheduled at the time slot, where just the majority of participants is available. Dudle (http://dudle.inf.tu-dresden.de), a web-application which implements the protocol is presented. We measured its performance in order to show that the protocol is practical and feasible.

Keywords: event scheduling, electronic voting, superposed sending, anonymity, privacy-enhanced application design.

1 Introduction

There are numerous Web 2.0 applications (e.g., doodle.com, moreganize.ch, ...), which allow users to create polls. The most important use case of these applications is to schedule events. They all have in common, that they disclose detailed *availability patterns* of their users. These patterns contain sensible information in at least two respects. First, one is able to read information *directly* out of such a pattern ("Does my boss work after 3pm?"). Secondly, due to the fact, that these patterns contain much entropy, one is able to connect them with other information sources easily to read *indirect* information and re-identify participants who would otherwise remain pseudonymous ("This availability pattern looks like Peters who goes to lunch everyday at 11:30."). All existing applications for event scheduling allow some privacy settings, but none of them tries to overcome the need of complete trust in the application server and the poll initiator.

A privacy-friendly and verifiable solution for scheduling a single event, which reveals only the sum of available participants at every time slot was proposed [1]. Despite being efficient enough for a Web 2.0-implementation, it has the drawback that unanimous agreement is required for resisting internal attacks.

In this paper we present, a better solution for the problem of internal attackers. Therefore we discuss two extensions to the original protocol described in [1]. These extensions prevent internal attacks and allow majority agreement simultaneously.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 235–246, 2011.

2 Related Work

There are several approaches dealing with event scheduling. It can be seen as distributed constraint satisfaction / optimization problem (DCSP/DCOP) or as an instance of electronic voting.

Many algorithms for DCSP [2,3,4] and DCOP [5,6,7] exist and measurements of the information leakage were done [8,9]. With the help of these algorithms, complex scheduling problems may be solved (e. g., scheduling of many events, where different subsets of the participants participate in each event with constraints about place, travel time etc.). However, all DCOP algorithms share the problem that they are complex in terms of message exchanges even for basic scenarios. To solve the problem of message exchanges, agents are used, which send and receive the messages. As users do not want to setup such an agent at some server, they have to run it locally and have to be online at the same time. Therefore, the DCOP approach is too complex in terms of message exchanges to be implemented in a web application and a simpler solution for the simpler problem of scheduling a *single* meeting would be appropriate.

There is a lot of literature about electronic voting [10,11,12,13,14,15,16], some of them lead to implementations [17,18,19]. 4 observations occur, when event scheduling is done with e-voting: (1) Only the sum of available participants, not the single availabilities, can be used to schedule the time slot. (2) The short ballot assumption [20] does not hold as the availability pattern contains much entropy and there are few voters (participants). (3) All participants have to be voting officials to minimize trust in other entities. (4) Coercion resistance is not necessarily required.

To overcome the short ballot assumption, one has to apply an e-voting scheme for every time slot, which is scheduled. However, if the computational complexity of the scheme depends on the number of time slots and the number of participants (voting officials, assumption 3), an application will exceed the possibilities of current browsers. To illustrate this, we implemented a performance measurement for a discrete exponentiation modulo a 786 bit long integer using different libraries and browsers. We measured 3 of the 5 investigated libraries. The remaining two were too slow to be mentioned here. If, e.g., a scheme would need only one asymmetric operation per time slot and participant and a concrete poll would ask for 20 time slots for a meeting with 5 participants, the scheme would need *only* for the asymmetric operations about $5 \cdot 20 \cdot 0.78\,\mathrm{s} = 78\,\mathrm{s}$, using the BigInteger Library of Wu on a Firefox 3.6, IE 8 would be blocked for at least 4 minutes. Having a low computation complexity is even more important considering small mobile devices like smartphones (cp. the last column of Table 1).

Besides e-voting, there are two specific schemes, which try to solve event scheduling in a privacy-friendly way [24,1], both offering unanimous agreement only. The main idea of [1] is to use DC-Net [25] to calculate the sum of available participants to the time slots. Therefore, a dedicated DC-Net round is executed for every nominated time slot. Each participant sends an encrypted 1 in the specific round if he is available at the time slot, and 0 otherwise. Through the homomorphism in the DC-Net, the sum of the votes is calculated. The result of one DC-Net round is the number of available participants at this time slot.

Table 1. Execution time for an exponentiation modulo a 786 bit long integer in JavaScript with different libraries. The first 5 columns were measured on an Intel Pentium 4 Duo with 2.8 GHz, 2 GB RAM running Windows XP SP3. The last one was measured on a Motorola Milestone running Android.

	IE 8.0.6001	Firefox 3.6.12	Safari 5.0.3	Opera 10.63	Chrome 8.0.552	Android Firefox 4.0b2
Wu [21]	2.80 s	0.78 s	0.91 s	0.31 s	0.10 s	7.87 s
Baird [22]	5.05 s	0.43 s	0.15 s	0.18 s	0.16 s	5.10 s
Shapiro [23]	21.47 s	2.12 s	1.18 s	0.99 s	0.61 s	(crashes)

The main problem of using the DC-Net is that a participant may send values different from 0 or 1. I.e., a participant may send values below 0 to lower the chance for a specific time slot of being chosen, and values above 1 to increase it. An example of this is illustrated in Fig. 1. The tables show two polls with 3 participants and 4 time slots $(t_0, \ldots, t_3)$. The unencrypted votes are shown inside each table ($v_{u,t} \in \{0, 1\}$). Mallory manipulates the poll in a way that time slot t_3 results in the largest sum. Because of the anonymization of all messages through the DC-Net, Mallory's attack is hidden to Alice and Bob.

In the following, we will discuss how to extend the original protocol [1] in a way that sending values different from 0 or 1 can be prevented without the need of unanimous agreement (which was used there to overcome the problem).

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	−1	−1	−1	1
$\sum$	0	1	−1	2

(a) attacking with −1

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	0	0	2
$\sum$	1	2	0	3

(b) attacking with +2

Fig. 1. Different ways to attack a poll if Mallory wants t_3 to win. The plain text votes $(v_{u,t})$ are displayed.

3 Preventing (−1)-Attacks

The main idea for preventing (−1)-attacks is the fact that the results of a DC-Net at some time slot must not be lower than 0. I.e., in Fig. 1a the attack cannot be detected at time slot t_0 and t_1, but it is visible at time slot t_2, where the result is −1. If all participants are honest, the result of every time slot should be a value between 0 and the number of all participants.

Instead of using one dedicated DC-Net round for every time slot, we use several simultaneously running DC-Net rounds. Let I be the number of simultaneously running DC-Net rounds (we will see later in this section, that I should

be chosen in some dependence of the number of participants). Every participant u splits each vote $v_{u,t} \in \{0,1\}$ into I partial votes $\bar{v}_{u,t,0}, \ldots, \bar{v}_{u,t,I-1}$ such that:

1. An index $j \in \mathbb{Z}_I$ for one partial vote is chosen randomly and kept secret.
2. The partial vote with index j ($\bar{v}_{u,t,j}$) is equal to the actual vote $v_{u,t}$.
3. The remaining $I-1$ partial votes are equal to 0.

Let T be the set of time slots, U the set of participants of the poll, σ_t the number of available participants at time slot t, $\bar{k}_{u,u',t,i}$ the DC-Key between two voters, and $\bar{d}_{u,t,i}$ the encrypted vote within a DC-Net round ($\bar{d}_{u,t,i} = \bar{v}_{u,t,i} + \sum_{u' \in U, u' \neq u} \bar{k}_{u,u',t,i}$). If all participants are honest, the following properties result from the construction:

1. For all time slots $t \in T$ and partial vote indices $i \in \mathbb{Z}_I$, the sum of all partial votes of all participants is an element between 0 and the number of participants ($\forall t, i : \sum_{u \in U} \bar{v}_{u,t,i} = \sum_{u \in U} \bar{d}_{u,t,i} \in \{0, \ldots, |U|\}$).
2. At one time slot, the sum of all partial votes of all participants is the sum of all available participants at this time slot ($\sigma_t = \sum_{u \in U, i \in \mathbb{Z}_I} \bar{v}_{u,t,i}$).

In Fig. 1, we have seen that an attacker has to guess at which time slot a honest participant will send a 1. With the proposed extension it is not sufficient for the attacker Mallory to guess the availability of another participant, she further has to guess at which partial vote the actual vote was sent. This is difficult as long as the chosen partial vote index is random, kept secret, and the number of partial votes per time slot $\mathbb{Z}_I$ is sufficiently high.

Fig. 2 shows an example of the vote vector splitting with $I = 3$, where Mallory tries to send a -1 at t_1 ($v_{u_m,t_1} = -1$). The left table shows the original protocol.

	t_0	t_1	t_2	t_3
Alice	0	1	0	0
Bob	1	1	0	1
Mallory	0	−1	0	1
$\sum$	1	1	0	2

	t_0	t_1	t_2	t_3	
Alice	0	1	0	0	$i = 0$
Bob	0	0	0	0	
Mallory	0	0	0	0	
$\sum$	0	1	0	0	
Alice	0	0	0	0	$i = 1$
Bob	0	1	0	1	
Mallory	0	0	0	1	
$\sum$	0	1	0	2	
Alice	0	0	0	0	$i = 2$
Bob	1	0	0	0	
Mallory	0	−1	0	0	
$\sum$	1	−1	0	0	
$\sum$	1	1	0	2	

Fig. 2. Split the votes into several partial votes. While Mallory's attack remains undetected in the left table, Alice and Bob are able to detect it in the right one.

There the attack would remain undetected as all elements of the result vector are in the allowed range. The right table shows the same vote vectors split into several vectors. There, for time slot t_1 Alice has chosen the first table ($i = 0$) for her vote ($\bar{v}_{u_a,t_1,0} = v_{u_a,t_1} = 1$) and Bob has chosen the second one. As Mallory has chosen the third table ($i = 2$) her attack can be detected.

Note, that the enrypted vote vectors must not be published until the last participant has sent his vector. Otherwise, if the server would cooperate with Mallory, she may wait until all other participants sent their vote and then calculate the preliminary result before casting her own vote. This allows her to state -1s where other participants sent a 1. As already stated in the original protocol, this restriction can be relaxed with one additional communication phase in which all voters have to commit to their votes [1].

3.1 Verifiability

When verifying that no attack occurred, we can distinguish two cases: (1) In the simpler case, one DC-Net round results -1 ($\sum_{u \in U} \bar{d}_{u,t,i} = -1$). This case occurred in the example in Fig. 2. (2) In a more complex scenario, some participant sent a 1 in some DC-Net round, but the result equals 0 due to a (-1)-attack. This may also occur in the original protocol. Bob for example can detect that Mallory has cheated at t_0 in Fig. 1a. However, to prove that Mallory has cheated at t_0, Bob has to give up his privacy. In such a case, Bob can decide for himself what is worth more, his privacy or to unmask Mallory.

We first discuss the case where privacy is the most valuable good, and we discuss situations later where participants are willing to give up their privacy for unmasking attackers. For reasons of simplicity, we write all calculations which are done in the DC-Net without the modulo operations. We consider only 1 attacker.

Without Privacy-Loss. Checking the correctness of a poll can be expressed by a function. It takes all messages sent within the poll and returns true if the poll was correct, and false otherwise $\mathcal{V} : \mathbb{Z}^{|U|\times|T|\times I} \rightarrow \{\text{true}, \text{false}\}$. Let $\bar{\boldsymbol{d}} \in \mathbb{Z}^{|U|\times|T|\times I}$ be the 3-dimensional array of all DC-Net messages containing elements of $\bar{d}_{u,t,i}$ for a DC-Net message from participant u at time slot t and partial vote index i. The function which checks the correctness of the poll is defined as:

$$\mathcal{V}(\bar{\boldsymbol{d}}) = \begin{cases} \text{true} & \text{if } \forall t \in T, i \in \mathbb{Z}_I : \sum_{u \in U} \bar{d}_{u,t,i} \in \{0, \ldots, |U|\} \\ \text{false} & \text{otherwise.} \end{cases} \tag{1}$$

We assume one attacker Mallory (u_m) who tries to send a -1 at time slot t. With an increasing amount of participants, voting for t, the probability of detection would decrease. In a worst case scenario w.r.t. detecting attackers all honest participants ($|U| - 1$) send a 1 for all time slots and therefore the lower bound of the probability to detect the attack is

$$P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \,|\, v_{u_m,t} = -1) \geq \left(\frac{I-1}{I}\right)^{|U|-1}. \tag{2}$$

The probability of successfully performing an attack would increase with an increasing number of participants. Therefore, one should choose the number of DC-Net rounds I dependent on the number of participants. If Mallory tries to send a -2 the chance of detection increases, but this is out of scope of this paper.

With Privacy-Loss. We already discussed that a person who sent a 1 in a DC-Net which results in 0 is in the position to unmask the attacker with the drawback of giving up his privacy.[1] For such a case, we can define another function checking the correctness of the poll. This function will return false if there exists a participant u_p, who can prove that he sent a 1 at a time slot t and DC-Net round with partial vote index i which resulted in 0 ($\sum_{u \in U} \bar{d}_{u,t,i} = 0$).[2] Let $\bar{\boldsymbol{k}} \in \mathbb{Z}^{|U| \times (|U|-1) \times |T| \times I}$ be the 4-dimensional array of all keys used in all DC-Nets of the poll. The function which checks the correctness of the poll under the assumption that all participants are willing to disclose their availability at one time slot to unmask an attacker is defined as

$$\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \begin{cases} \text{true} & \text{if } \neg\exists u_p \in U, t \in T, i \in \mathbb{Z}_I : \\ & \left(\sum_{u \in U} \bar{d}_{u,t,i} = 0\right) \wedge \left(\bar{d}_{u_p,t,i} + \sum_{u \in U, u \neq u_p} \bar{k}_{u_p,u,t,i} = 1\right) \\ \text{false} & \text{otherwise.} \end{cases} \tag{3}$$

Now, Mallory needs two honest participants sending a 1 in the same DC-Net round to hide her (-1)-attack under these assumptions.[3] The probability that this attack is detected is calculated by adding the probabilities of the two cases (1) nobody choses Mallory's DC-Net, and (2) one participant choses Mallory's DC-Net. For case 1, the sum of the attacked DC-Net will be -1 and therefore $\mathcal{V}(\bar{\boldsymbol{d}})$ will fail (see Equation 2). The lower bound[4] for the probability of case 2 is the probability where the output of $\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right)$ is false and can be calculated with

$$P(\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \text{false} \,|\, v_{u,t} = -1) \geq (|U| - 1) \cdot \frac{1}{I} \cdot \left(\frac{I-1}{I}\right)^{|U|-2}. \tag{4}$$

Note that it is not possible, that $\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false}$ and $\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \text{false}$ (cp. Footnote 2) if we stick to only one attack at one time and therefore we can add the probabilities of Equations 2 and 4 to get the overall probability of detecting a (-1)-attack if users are willing to disclose their availability to unmask attackers.

Summary. Table 2 illustrates these formulas with some example values. One can see that splitting the vote vector into 20 partial vote vectors makes it rather unlikely to perform an undetected attack against small polls with 5 participants.

[1] E. g., Bob could detect that Mallory cheated at t_0 in Fig. 1a.

[2] If this sum is lower than 0, an attack occurred as well. However, as this attack would be discovered by function $\mathcal{V}(\bar{\boldsymbol{d}})$ (Equation 1), we want to neglect this case here.

[3] This is like trying to perform a (-2)-attack without privacy-loss, but choosing one partial DC-Net to send the -2.

[4] All $|U| - 1$ honest participants voted for the attacked time slot.

Table 2. Lower bounds for the probability of successfully detecting an attack

I	$\lvert U\rvert$	(a)	(b)	(c)
20	15	48.8 %	84.7 %	(35.9 %)
20	5	81.5 %	98.6 %	(17.1 %)
50	15	75.4 %	96.9 %	(21.5 %)
50	5	92.2 %	99.8 %	(7.5 %)
100	15	86.9 %	99.2 %	(12.3 %)
100	5	96.1 %	99.9 %	(3.9 %)

(a) without giving up privacy $P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \mid v_{u,t} = -1)$
(b) with privacy-loss $P(\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false} \vee \mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \text{false} \mid v_{u,t} = -1)$
(c) probability of privacy-loss $P(\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right) = \text{false} \mid v_{u,t} = -1)$

The chance to detect a (-1)-vote at each time slot is at least[5] 81.5 %. Additionally to these 81.5 %, the attack can be discovered with a probability of 17.1 % by one of the participants. If all participants are willing to disclose their availability at the attacked time slot, the detection probability is at least 98.6 %.

In case of $\mathcal{V}(\bar{\boldsymbol{d}}) = \text{false}$, the decryption of the DC-Net round can be requested where the invalid value occured. Therefore every participant has to reveal his key for the DC-Net round. The single votes can be decrypted with the keys which identifies the attacker. The attacker may modify her key to hide her attack in this phase. However, this can be prevented in the same way as it was proposed for the verification phase of the original protocol.

However, if availabilities should not be disclosed under any circumstances, the attacker identification phase may be skipped. One has to accept in this case that attackers are able to perform denial of service attacks anonymously. Then one may decide with function $\mathcal{V}(\bar{\boldsymbol{d}})$ (Equation 1) that some attack occurred, but skip revealing keys to avoid possible decryption of votes. Note that the decision to perform a poll with identification phase or without has to be accepted by all participants. If every participant may decide on his own, an attacker will always refuse to reveal her keys, stating she has to cover some vote.

If a participant discovers an attack with the function $\mathcal{B}\left(\bar{\boldsymbol{d}}, \bar{\boldsymbol{k}}\right)$, he may decide on his own if he gives up his privacy to unmask the attacker. Therefore, the lower bound for the probability of detecting an attack is between both lower bounds.

3.2 Privacy

The possible decryption in the identification phase to detect the attacker may be a privacy problem. The original protocol had the same decryption in the verification phase. However, unlike in the original protocol, the probability that this really is a privacy problem is very low, as the attacker has to guess the index for the DC-Net round where the victim sent his vote. The probability of guessing the index of a specific victim is $\frac{1}{I}$.

Attacking more than one DC-Net round with negative values to increase the probability of hitting the victim's DC-Net does not help the attacker, because the goal of the honest participants is to find only one DC-Net which was attacked. Therefore, it is enough to disclose the keys for one attacked round. The algorithm

[5] if all 4 honest participants vote for a time slot.

to choose the DC-Net round to be disclosed should choose one of the rounds with the lowest sum. The function $\mathcal{D} : \mathbb{Z}_n^{|U|\times|T|\times I} \to \mathcal{P}(T \times \mathbb{Z}_I)$ which takes all DC-Net messages as input, and results a set of time slot-partial vote index-pairs (t, i) which should be disclosed, can be defined as

$$\mathcal{D}(\boldsymbol{d}) = \left\{ (t,i) : \sum_{u\in U} \bar{d}_{u,t,i} = \min_{t'\in T, i'\in\mathbb{Z}_I} \left\{ \sum_{u\in U} \bar{d}_{u,t',i'} \right\} \right\}. \tag{5}$$

However, as already discussed in Section 3.1, one may decide to skip attacker identification, with the drawback, that denial of service attacks are possible then.

The privacy of a message in the DC-Net depends only on the secrecy of the keys. This is the same for the original as well as the new protocol but splitting the vote vector into several parts introduces a new point of attack. Now, the anonymity of the message also depends on the randomness and secrecy of the partial vote index. If an attacker can predict at which DC-Net rounds messages from some participants occur, she can separate the other messages into smaller anonymity sets. If all participants distribute their votes randomly over all rounds, Mallory needs the cooperation of the other participants to deanonymize her victim. However, deanonymization can be done without the help of the different partial DC-Nets, if participants disclose their shared DC-Net keys.

In a successful schedule with no attacker, one sum for every time slot with the number of all available participants is disclosed.

4 Preventing (+2)-Attacks

In the example of Fig. 1b, Alice can detect a (+2)-attack of Mallory as she knows that the sum is an element of $\{0, \ldots, |U| - 1\}$. However, it may be the case that nobody voted for a time slot (cp. e.g., time slot t_2 of Fig. 1b) where a 2 is sent. In such a case neither Alice nor Bob can detect the attack on their own.

A simple solution to this attack would be to request the verification phase in any case for the agreed time slot. This would neither be privacy-friendly nor efficient in terms of message exchanges.

In the following, another solution is proposed. The main idea is to reduce the problem of preventing (+2)-attacks to the already solved problem of preventing -1-attacks. Therefore, in addition to the normal poll, every participant sends his votes for the same time slots in a check poll. Every participant u calculates for every time slot t a check vote $v'_{u,t}$ which depends on his vote $v_{u,t}$ such that

$$v_{u,t} + v'_{u,t} = 1. \tag{6}$$

With the check votes $v'_{u,t}$, a check poll is done like for the normal poll.

The sum of both result vectors, the one from the normal poll and the one from the check poll, should be a vector where all elements are equal to the number of participants $|U|$. By checking this property, it is ensured that every participant calculated the check vote vector according to Equation 6. If Mallory wants to send a 2 for some time slot, she then has to send a -1 in the check poll.

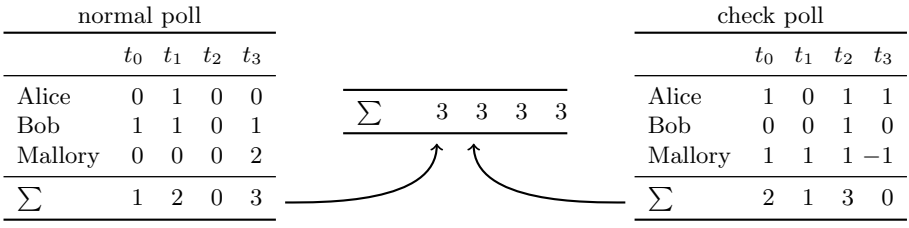


Fig. 3. By the use of a check poll, the (+2)-attack can be reduced to the (−1)-attack. Mallory has to send a −1 at t_3 in the right table, because the sum regarding t_3 of both tables would not be equal to the number of participants otherwise.

However, splitting the check vote vector into several ones, this attack can be prevented in the same way (−1)-attacks were prevented within the vote vector. Fig. 3 illustrates the whole process.

Let $\bar{\boldsymbol{d}}$ be the 3-dimensional array of all DC-Net messages sent to the normal poll and $\bar{\boldsymbol{d}}'$ be the 3-dimensional array of all DC-Net messages sent to the check DC-Nets. The verification function of correctness is defined as

$$\mathcal{C}(\bar{\boldsymbol{d}}, \bar{\boldsymbol{d}}') = \begin{cases} \text{true} & \text{if } \forall t \in T : |U| = \sum_{u \in U, i \in \mathbb{Z}_I} \left(\bar{d}_{u,t,i} + \bar{d}'_{u,t,i}\right) \\ \text{false} & \text{otherwise.} \end{cases} \tag{7}$$

4.1 Verifiability

When evaluating the result, two kinds of inconsistencies may occur: the sum of both polls may be lower or higher than the number of participants. As we split the votes into several ones (cp. Section 3), we do not consider (−1)-attacks at this point. Therefore, a value lower than the number of participants may occur only if one or more participants sent $v_{u,t} = v'_{u,t} = 0$. Having such a case would mean that the number of available participants σ_t would be the result of the normal poll at least. The difference of the sum of both polls and the total number of participants are wrongly cast votes.

The second kind of inconsistency is if the sum of both polls is higher than the number of participants. As we prevented (−1)-votes, the result of the normal poll at a time slot t should be greater or equal than the number of available participants at this time slot. In addition, the number of available participants is greater or equal than the total number of participants minus the result of the check poll at a certain time slot. Putting both inequations together, one obtains a range which expresses the possible number of available participants:

$$\sum_{i \in I, u \in U} \bar{d}_{u,t,i} \geq \sigma_t \geq |U| - \sum_{i \in I, u \in U} \bar{d}'_{u,t,i}. \tag{8}$$

However, as an attacker may manipulate his vote in a way that this range results in $\{0, \ldots, |U|\}$, he may attack the availability of the poll in such a way. To unmask the attacker, all DC-Net rounds for the inconsistent time slot can be decrypted as shown before. If the attack discovery goes along with some cost (penalty, reputation loss, etc.), it makes such attacks unattractive.

4.2 Privacy

During the verification phase of an inconsistent check poll, the availability of all participants at the inconsistent time slot are disclosed. To avoid disclosure of all availabilities, one may disclose the DC-Net rounds step by step and stop when the attacker is found. The sequence of disclosure should therefore be a fixed order, which is not known before every participant stated his vote.[6]

In a successful run, no more information is disclosed than in the original protocol, the check poll contains only redundant information.

4.3 Computational Complexity

The extension presented in this paper affects the computational complexity of the original protocol only in terms of symmetric cryptographic operations, i.e., the amount of asymmetric cryptographic operations is not affected. The number symmetric operations of the original scheme increases with the number of DC-Net rounds I and is doubled due to the check poll. Table 3 compares the complexity of our extension with the complexity of the original protocol.

Table 3. Comparison of the computational complexity of the original protocol with unanimous agreement and the scheme with our extension (assuming no attack)

	discrete exp.	symmetric decr.	hash values
original protocol	$\|U\| - 1$	$\|T\| \cdot (\|U\| - 1)$	$\|T\| \cdot (\|U\| - 1)$
new scheme	$\|U\| - 1$	$2 \cdot I \cdot \|T\| \cdot (\|U\| - 1)$	$2 \cdot I \cdot \|T\| \cdot (\|U\| - 1)$

5 Implementation

An implementation of the protocol is available at `dudle.inf.tu-dresden.de`. The cryptographic operations are implemented in JavaScript; no installation on the client side is needed.

The implementation has been done using the JavaScript BigInteger library from Tom Wu [21]. Like in the original protocol, a symmetric cipher and a hash function is used for key generation. AES-128 and SHA-256 from the JavaScript libraries of B. Poettering are used here [26].

Table 4 shows a performance measurement of the key calculation for an example poll. One can see, that the calculation needs about 23 s, using Firefox 3.6.12. If browsers run a script which needs longer calculation time, it is usual that the browser asks the user if he wants to stop the script. To avoid these pop-ups, the BigInteger library was modified in a way that it calculates exponentiations asynchronously with a callback function. This enables the calculation to be forked in

[6] E.g., every participant may commit himself to a random number together with his vote vector. In case of verification all commitments are revealed and the random numbers are added to one single seed which is used to bootstrap a sequence.

Table 4. Performance measurement of the key calculation in a poll with $|U| = 5$, $|T| = 20$ and $I = 20$. The first 5 columns were measured on an Intel Pentium 4 Duo with 2.8 GHz, 2 GB RAM running Windows XP SP3. The last one was measured on a Motorola Milestone running Android.

	IE 8.0.6001	Firefox 3.6.12	Safari 5.0.3	Opera 10.63	Chrome 8.0.552	Android Firefox 4.0b2
AES-128+SHA-256	18.4 s	7.7 s	2.9 s	1.4 s	2.8 s	31.7 s
DH	15.0 s	11.7 s	8.2 s	2.0 s	2.5 s	40.5 s
total	37.6 s	22.5 s	13.5 s	4.3 s	6.3 s	84.2 s

the background and the browser remains responsive. The user can enter his availabilities, while the browser calculates the keys. The submit button is enabled after the calculation has been done. Assuming, that a user needs some time to enter his preferences (look up the time slots in his personal calendar, click the buttons etc.), there is no extra waiting time.

6 Conclusion

We proposed a scheme, which is able to schedule events in a privacy-enhanced way. Our scheme prevents attacks to neglect or promote certain time slots, has no negative influence on the privacy and affect the computational complexity only in terms of symmetric cryptographic operations. To demonstrate that the scheme performs in practice, we presented Dudle, an implementation of the scheme in JavaScript. Due to the use of JavaScript for all client side operations, no installation is needed for the user. We therefore showed, that complex cryptography is possible in zero footprint applications.

Acknowledgments. The author wants to thank the whole DuD-Group in Dresden for the fruitful coffee breaks. Special thanks goes to Rainer Böhme, Sebastian Clauss, Stefan Köpsell and Sandra Steinbrecher. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement № 216483.

References

1. Kellermann, B., Böhme, R.: Privacy-enhanced event scheduling. In: CSE, vol. 3, pp. 52–59. IEEE Computer Society, Los Alamitos (2009)
2. Silaghi, M.C., Sam-Haroud, D., Faltings, B.: Asynchronous search with aggregations. In: AAAI/IAAI, pp. 917–922. AAAI Press / The MIT Press (2000)
3. Yokoo, M., Hirayama, K.: Algorithms for distributed constraint satisfaction: A review. Autonomous Agents and Multi-Agent Systems 3(2), 185–207 (2000)
4. Léauté, T., Faltings, B.: Privacy-preserving multi-agent constraint satisfaction. In: CSE, vol. 3, pp. 17–25. IEEE Computer Society, Los Alamitos (2009)

5. Modi, P.J., Shen, W.M., Tambe, M., Yokoo, M.: Adopt: Asynchronous distributed constraint optimization with quality guarantees. Artif. Intell. 161, 149–180 (2005)
6. Maheswaran, R.T., Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P.: Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In: AAMAS, pp. 310–317. IEEE Computer Society, Los Alamitos (2004)
7. Mailler, R., Lesser, V.: Solving distributed constraint optimization problems using cooperative mediation. In: AAMAS, pp. 438–445. IEEE, Washington, DC (2004)
8. Franzin, M.S., Freuder, E.C., Rossi, F., Wallace, R.: Multi-agent meeting scheduling with preferences: Efficiency, privacy loss, and solution quality. AAAI Technical Report (2002)
9. Greenstadt, R., Pearce, J.P., Bowring, E., Tambe, M.: Experimental analysis of privacy loss in DCOP algorithms. In: AAMAS, pp. 1424–1426. ACM Press, New York (2006)
10. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–90 (1981), doi:10.1145/358549.358563
11. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
12. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: STOC, pp. 544–553. ACM, New York (1994)
13. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
14. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
15. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
16. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES, pp. 61–70. ACM, New York (2005)
17. Herschberg, M.A.: Secure electronic voting over the world wide web. Master's thesis, Massachusetts Institute of Technology (May 1997)
18. Adida, B.: Helios: Web-based open-audit voting. In: USENIX, pp. 335–348 (2008)
19. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: Security and Privacy, pp. 354–368. IEEE, Los Alamitos (2008)
20. Rivest, R.L., Smith, W.D.: Three voting protocols: Threeballot, vav, and twin. In: USENIX, p. 16. USENIX Association, Berkeley (2007)
21. Wu, T.: BigIntegers and RSA in JavaScript (July 2010), `http://www-cs-students.stanford.edu/~tjw/jsbn/`
22. Baird, L.: BigIntegers in JavaScript, Version 5.4. (July 2010), `http://www.leemon.com/crypto/BigInt.html`
23. Shapiro, D.: BigInt, a suite of routines for performing multiple-precision arithmetic in JavaScript, Version 5.4. (July 2010), `http://ohdave.com/rsa/BigInt.js`
24. Herlea, T., Claessens, J., Preneel, B., Neven, G., Piessens, F., De Decker, B.: On securely scheduling a meeting. In: SEC. IFIP Conference Proceedings, vol. 193, pp. 183–198. Kluwer, Dordrecht (2001)
25. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1(1), 65–75 (1988)
26. Poettering, B.: The AES block cipher and the SHA256 message digest in JavaScript, Version 0.1. (July 2010), `http://point-at-infinity.org/`

Analyzing Key-Click Patterns of PIN Input for Recognizing VoIP Users

Ge Zhang

Karlstad University, Universitetsgatan 2,
65188, Karlstad, Sweden
ge.zhang@kau.se

Abstract. Malicious intermediaries are able to detect the availability of VoIP conversation flows in a network and observe the IP addresses used by the conversation partners. However, it is insufficient to infer the calling records of a particular user in this way since the linkability between a user and a IP address is uncertain: users may regularly change or share IP addresses. Unfortunately, VoIP flows may contain human-specific features. For example, users sometimes are required to provide Personal identification numbers (PINs) to a voice server for authentication and thus the key-click patterns of entering a PIN can be extracted from VoIP flows for user recognition. We invited 31 subjects to enter 4-digital PINs on a virtual keypad of a popular VoIP user-agent with mouse clicking. Employing machine learning algorithms, we achieved average equal error rates of 10-29% for user verification and a hitting rate up to 65% with a false positive rate around 1% for user classification.

1 Introduction

Current Internet users heavily rely on distributed networking intermediaries to transmit packets. These networking intermediaries might be compromised and thus cannot be simply trusted by users. Malicious intermediaries can wiretap their relayed packets for man-in-the-middle attacks. This threat is also an issue for Voice over IP (VoIP) services. Previous research [28] shows that it is easy for an intermediary to detect the availability of VoIP conversation flows between two hosts without reading the flow details. This actually reveals the VoIP calling records which include the IP addresses used by the conversation partners, the starting and ending time of the conversations. Other work [27,26] proposed more advanced method using watermark to increase the robustness and accuracy of the calling records detection. Calling records could reveal daily life of a user. For instance, spammers may infer the requirements and preference of a particular user from the calling records (e.g, a recent calling record showing that a user calls a dentist reveal that the user might have dental problems) so that they can send advertisements more effectively. Recently news report that third parties offer traditional telephone calling records for profit [1]. With the increasingly deployment and usage of VoIP services, it can be predicted that the confidentiality of calling records will be an important issue on VoIP as well.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 247–258, 2011.

Nevertheless, previous VoIP tracking methods [27,26] at most enable intermediaries to find out the calling records between two hosts identified with their IP addresses (IP-level calling records). The linkability between a VoIP user and a IP addresses is usually unstable: VoIP users may move from one network to another network with their laptops, or switch between devices (e.g., the user switches from the home computer to an office workstation). Therefore, a user may use different IP addresses at different times. In addition, even one IP address might be shared by several users due to current limited IP address space [18]. Thus, the IP-level calling records are not accurate enough to attack a particular user. In this case, attackers require user-level calling records. To solve this, attackers need to extract human-specific characteristics from flows for user recognition. Previous papers [20] [8] verified that speech features can be extracted to re-identify a user if the user employs a specific codec. This paper investigate another alternative by taking advantage of user key-click patterns: Some automated voice services require users to provide their Personal identification numbers (PINs) for authentication (e.g., access a voice mailbox or a configuration setup). In this situation, users enter their PINs on their VoIP user-agents so that the user-agents generate specific packets to indicate which keys have been clicked. Thus, the key-click pattern for PIN input is a potential characteristic for user recognition. We addresses the following research questions in this paper: (1) How can a malicious intermediary recover the key-click patterns from intercepted VoIP flows? (2) How to minimize the impact from networking conditions (e.g., jitter, packet loss)? (3) Is the recovered key-click patterns accurate enough for user recognition? To answer these questions, we invited 31 subjects to participate in the experiments. Each of them entered 4-digital PIN codes on a popular VoIP user-agent by mouse clicking. Employing machine learning algorithms, we achieved average equal error rates of 10-29% for user verification and a hitting rates up to 65% with a false positive rate around 1% for classification. Finally, we also discuss corresponding countermeasures to prevent user recognition.

The rest of this paper is organized as follows. Section 2 introduces some background information about VoIP. Section 3 introduces the general idea of the attack method. Section 4 presents the preparation, procedure and results of the experiments that we conducted. Section 5 lists related work. Finally, we summarize this paper in Section 6.

2 Background in VoIP Flows

The Realtime Transport Protocol (RTP) [24] standardizes the packet format for VoIP conversations. RTP provides end-to-end delivery schemes for data with real-time characteristics over IP networks. It supports a variety of payload types, two of which are especially related to this paper.

– Voice payload: In a conversation, a user-agent constantly encodes acoustic signal into digital data as voice payloads using a codec. The user-agent on

the other side recovers the acoustic signal by decoding the payloads from the received RTP packets. We name this kind of RTP payloads as *RTP voice packets*. In many cases, a user-agent continuously generates RTP voice packets at a constant time interval (e.g., 20 ms) in a conversation unless other types of RTP packets with higher priority are triggered.

– Event payload [25]: When a user clicks a phone key in a conversation, the user-agent generates RTP packets with event payloads to indicate which key has been clicked. The RTP packets with event payloads are called *RTP event packets*. RTP event packets have higher priority than RTP voice packets.

The Secure Realtime Transport Protocol (SRTP) scheme [10] has been widely applied to protect RTP packet payloads by encrypting. However, it does not protect RTP headers[1]. This means that RTP header fields are available to intermediaries despite of the protection. Several RTP header types are introduced as follows:

– Marker bit: It indicates the beginning of a new event. For instance, a user presses a key "4" may span a series of RTP event packets, but only the first packet has the marker bit set. We define that a RTP event packet with marker bit set is a *key-down* packet and the others are *key-holding* packets.
– Payload type: It identifies the type of the RTP payload.
– Sequence number: The RTP sequence number is incremented by one in each successive RTP packet sent. The sequence numbers are assigned to RTP packets to allow the receiver to restore the original sequence in case of unreliable transmission.

Figure 1 plots a typical RTP flow with both event and voice packets. The packet inter-arrival time is around 20 ms. It can be predicted that 4 events are represented in this flow. Each event contains 1 key-down packet and a set of key-holding packets.

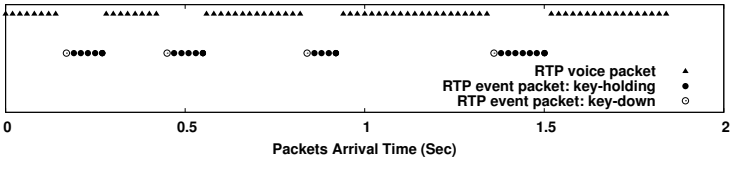
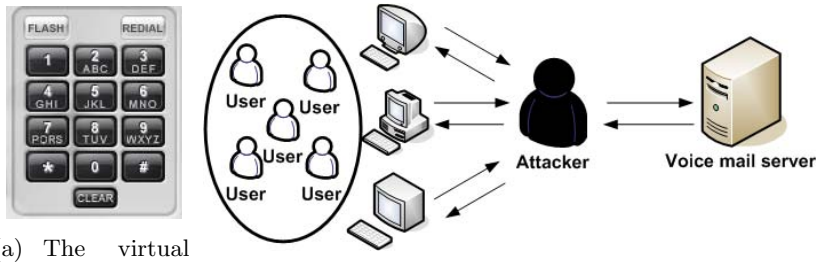


Fig. 1. The packet inter arrival time of an example RTP flow

[1] RTP headers are sent in the clear to allow for billing purposes and header compression.

3 Attacking Method

Image the following scenario illustrated in Figure 2(b): There are several users whose user-agents share the same IP address. An attacker is a malicious intermediary which intercepted a number of RTP flows originated from this IP address. The packets arrival time in the flows are recorded. Some flows have been already correctly labeled with their originator users (we call these flows as *testing flows*). The rest without being labeled are called *testing flows*. The attacker aims to further profile user-level calling records using the testing flows. Thus, the attacker may want to (1) label the flows for a particular user; or (2) label the flows for all users. We assume that the users occasionally access their voice mail by providing their PINs on a virtual keypad of a user-agent using a mouse[2] (see Figure 2(a)). In addition, all RTP flows in the environment are encrypted by using SRTP [10].



(a) The virtual keypad

(b) The network topology

Fig. 2. The environment of experiments

If the attacker detects a VoIP flow for PIN input and recognizes its originator user, it is highly possible that the calls at the time around are done by the same user. In this way, we consider to recognize a user by key-click patterns. The challenge is how to recover the key-click patterns from RTP flows? Given an encrypted RTP flow, the attacker firstly needs to distinguish the RTP event packets and RTP voice packets. Actually it is rather simple: Despite of the protection by SRTP, the RTP headers are still in plain text. Thus, the attacker can distinguish them directly by reading the "Payload-Type" header fields. After picking RTP event packets from a flow, the next step is to restore the key-click behavior. To input 4-digital PIN code, 4 key-click events are generated. The attack can observe 4 key-down packets with following key-holding packets from the marker bits in headers. Moreover, we define the last key-holding packet for each event as the *key-up* packet. The attacker can restore a key-click pattern by guessing 4 key-holding periods (The period between a key-down packet and its following key-up packet) and 3 key-switching periods (the period between a key-up packet and the next key-down packet). Let us take the example flow in

[2] Some user-agents may also support keyboard input, but in this paper we only consider virtual keypad input.

Figure 1, the 4 key-holding periods are: 0.17-0.28s, 0.45-0.58s, 0.85-0.92s and 1.3-1.5s. The 3 key-switching periods are 0.28-0.45s, 0.58-0.85s and 0.92-1.3s.

The 4 key-holding periods and 3 key-switching periods are taken as variables for training and testing. Then, the attacker can employ a learning algorithm to construct a classifier, which builds key-click pattern using the training data and classifies the testing flows into correct classes. We did several experiments and the detailed work will be introduced in the next section.

Nevertheless, current Internet does not guarantee the quality of packet transmitting. Since this attack needs exact inter-packet arrival time of RTP event packets, the varying network quality (namely **jitter** and **packet loss**) could lead to an inaccurate observation. (1) Jitter indicates latency variations for different packets in a flow. For instance, a packet sent earlier may arrive the destination later. A large jitter on RTP event packets could make the key-click pattern recovering unreliable. Nevertheless, the sequence number on packet header field can help attackers to restore the original packet sequence. Moreover, attackers know what the fixed time interval between two successive packets should be (e.g., 20 ms). In this way, attackers can restore the packet inter-arrival time and sequence. Thus, the impact of jitter is not vital. (2) Packet loss indicates the amount of packets which are accidentally dropped in the transmission. Although attackers can detect packet loss rate by reading sequence number, they do not know the type of the lost packet for key-click pattern recovery. However, the attacker can heuristically guess it by the types of its neighbor packets. For example, if the lost packet is a key-holding packet in the middle, it is also easy for attackers to guess since the packets before and after are the same type.

4 Experiments

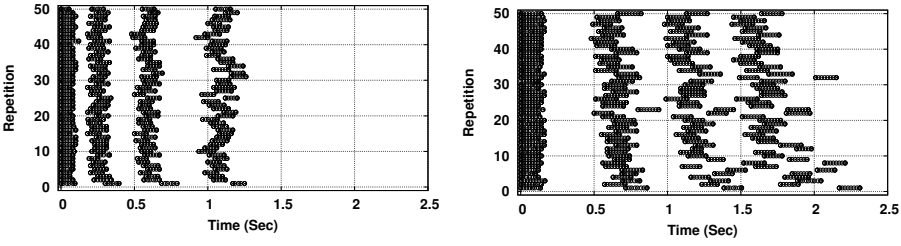
4.1 Data Collecting

To test the performance of this kind of attack, we did a series of experiments. We invited 31 students as test subjects who are denoted by $S = \{s_1, s_2, \cdots, s_{31}\}$. All of them have experience with using a computer, a mouse and a VoIP client. Each subject was asked to input two kinds of PIN codes. First, we randomly generated 31 different 4-digital PIN codes and thus assigned these PIN codes to the subjects one by one. Each subject was asked to input his/her unique PIN code using the mouse on the virtual-keypad of the X-Lite [7], a popular user-agent for 50 repetitions. We call these repetitions as *unique input repetitions* and let $du(s_i)_j$ to denote repetition j done by subject s_i. Furthermore, some users may have the same PIN code since there is only a 10^4 space for a 4-digital PIN code. Taking this into account, we arbitrarily selected a particular PIN code ("9913") and again asked each subject to input it for 50 repetitions. Similarly, we call the repetitions as *a shared input repetitions* and use $ds(s_i)_j$ to denote these repetitions.

We employed two computers: one ran X-Lite (version 3.0) as the user-agent. It is equipped with a DELL 19-inch flat panel LCD screen with 1024x768 pixel resolution. The mouse is a HP USB optical wheel mouse with the default speed setup on Windows XP platform. Another ran TCPDump [5] to simulate attackers to intercept RTP flows.

4.2 Data Processing

Following the method introduced in Section 3, we first extract RTP event packets from each flow. Figure 3(a) and Figure 3(b) illustrate the arrival time of RTP event packets of the unique input repetitions done by s_{12} and s_{17}. ($du(s_{12})_j$ and $du(s_{17})_j$, $1 \leq j \leq 50$). At a glance, readers can find the general key-click patterns are different for the two users. Then, we restore the 4 key-holding periods and 3 key-switching periods for each repetition using the method introduced in Section 3.



(a) $du(s_{12})_j$, $1 \leq j \leq 50$ (b) $du(s_{17})_j$, $1 \leq j \leq 50$

Fig. 3. Generated RTP event packets of the unique input repetitions done by s_{12} and s_{17}: one dot indicates one RTP event packet and the x-axis indicates packets inter-arrival time

4.3 Learning Algorithms

We employ three popular learning algorithms in this paper since these algorithms have been widely tested and performed well in previous work on keystroke biometrics [23,22]:

- Supporting Vector Machine (SVM) [11]: SVM works by constructing an N-dimensional hyperplane that groups the data into two spaces. In principle, it only solves the two-class problems. For multi-class problems, the algorithm can repeatedly perform the operations over all the possible two-class pairs and then find the suspect class by a voting mechanism.
- Random Forest (RF) [12]: Random forest is an ensemble learning method by generating a large number of bootstrapped classification trees and aggregating them during the training. Different to SVM, random forest can perform variable selection by itself and thus it is robust against noise. In a previous comparison, random forest provides a better predictive accuracy than other learning algorithms [14].
- Recursive partitioning (RPart) [13]: Recursive partitioning is a tree-based method for classifying data. It creates a classification tree and further splits the tree based on the condition of variables. The split process will be repeated for each leaf node until a certain stop splitting condition is met. Recursive partitioning is a fast classification algorithm.

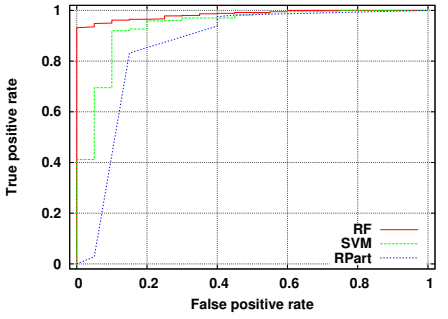
We implemented the classifiers in the R platform (version 2.12.0) [6], which provides a wide variety of statistical functions including classification based on the S language. In this paper, we implemented our classifiers using e1071 (SVM) [2], random forest [3] and recursive partitioning [4] packages. By using these packages, we can focus on our classification problems rather than the detail implementation of the algorithms.

4.4 Analysis and Results

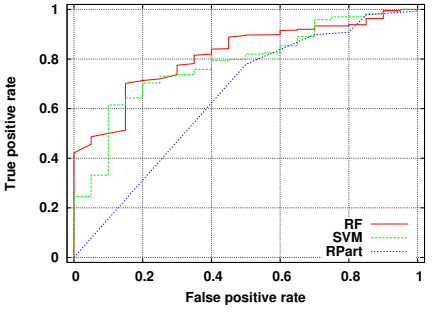
We recovered the 7 variables (4 key-holding periods and 3 key-switching periods) from each repetition. For each subject, we selected the first 30 repetitions ($du(s_i)_j$ and $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 1 \leq j \leq 30$) for training and took the rest 20 repetitions ($du(s_i)_j$ and $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 31 \leq j \leq 50$) for testing. In this paper, we focus on two problems, namely user verification and user classification.

- User verification: Given a testing repetition and a specific user s_x, the attacker asks the classifier whether the testing repetition was done by s_x. Thus, it is a binary classification problem (the real user s_x and imposters $s_{i \neq x}$). In this way, we split the training repetitions into 2 classes. One contains the repetitions done by user s_x ($du(s_i)_j$ or $ds(s_i)_j$, $i = x \wedge 31 \leq j \leq 50$) and another contains all the remaining repetitions ($du(s_i)_j$ or $ds(s_i)_j$, $i \neq x \wedge 31 \leq j \leq 50$). Given the testing repetitions ($du(s_i)_j$ or $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 31 \leq j \leq 50$), the classifiers calculate the scores showing how likely these repetitions were done by the user s_x. Finally, attackers can set a *decision threshold* to distinguish the real user and the imposters.
- User classification: In this case, there are 31 classes, each of which represents one subject. The attacker trains the classifiers with the training repetitions which have been correctly labeled their classes. Given the testing repetitions ($du(s_i)_j$ or $ds(s_i)_j$, $1 \leq i \leq 31 \wedge 31 \leq j \leq 50$), the classifiers distinguish them into the 31 classes using the default decision threshold. Finally, we create a 31 by 31 dimensional confusion matrix in which the element in row s_i, column s_j is a count of the number of times the subject with true ID s_i was classified into ID s_j.

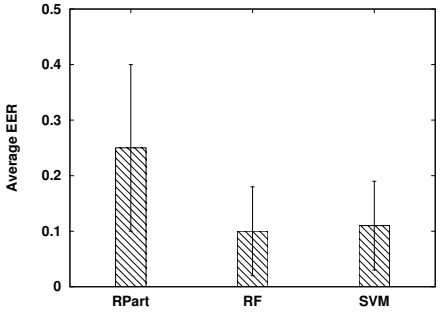
Like most previous work on classification problems, we evaluate the performance of the implementation by the classification errors, in more detail, false positive, which mistakenly takes the imposter's data as the real user's; and false negative, which mistakenly classifies the real user's data into the imposter's class. For evaluating the user verification problem, we also concern the Equal Error Rate (EER). In a binary classification problem, false positive rate and false negative rate usually varies depending on the decision threshold. EER, as the crossover point at which the false positive rate equals the false negative rate, is an important value to judge the classifier. The lower the EER, the better performance for the classifier. We run the implementation of our classifiers several times for the experiments. Figure 4 shows the result.



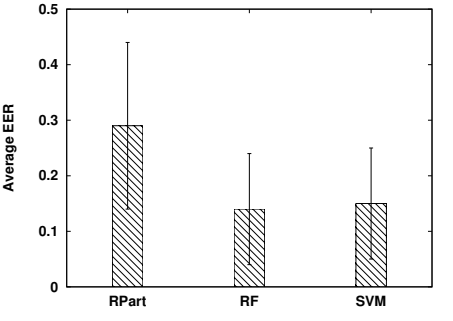
(a) The ROC curve for verifying s_{10} (unique input repetitions)



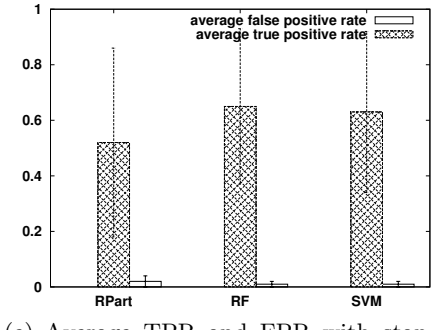
(b) The ROC curve for verifying s_{10} (shared input repetitions)



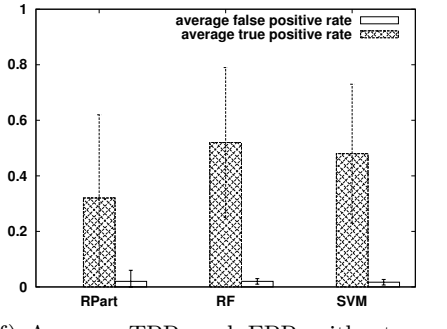
(c) Average EER with standard deviation for verifying $s_i, 1 \leq i \leq 31$ (unique input repetitions)



(d) Average EER with standard deviation for verifying $s_i, 1 \leq i \leq 31$ (shared input repetitions)



(e) Average TPR and FPR with standard deviation for classifying (unique input repetitions)



(f) Average TPR and FPR with standard deviation for classifying (shared input repetitions)

Fig. 4. Statistical results of our experiments on verification and classification using the 3 algorithms

Figure 4(a) and 4(b) illustrate the Receiver Operating Characteristic (ROC) curves with subject s_{10} as the genuine user when we did user verification. The repetitions are $du(s_{10})_j$ and $ds(s_{10})_j$ respectively. The True Positive Rate (TPR) is the frequency with the repetitions of subject s_{10} has been correctly detected. The False Positive Rate (FPR) is the frequency with which the imposters are mistakenly detected as the genuine users. Both of them varies depending on the decision threshold. We observe that the RF gives the best result and the RPart gives the worst result for both unique and shared input repetitions. The EER in unique input repetitions is from 0.08 to 0.18 and that in shared input repetitions is from 0.3 to 0.4. Figure 4(c) and 4(d) show that the average EER with standard deviation for all the subjects. The RF gives the average EERs around 0.1 and 0.14 for unique and shared input repetitions. The EERs given by SVM are 0.12 and 1.5 respectively. RPart gives the highest EER, around 0.25 and 0.29. As said, the lower the EER, the better performance for the classifier. Therefore, RF gives the best result. Figure 4(e) and 4(f) show the result of the classification problem. The best performance is still given by RF, with the lowest average TPR around 0.65 and FPR around 0.01 for unique input repetitions. The TPR for unique and shared input repetitions are 0.52 with 0.02 FPR. SVM has the similar results to RF. The worst case is still given by the RPart: its average TPR is around 0.52 for unique input repetitions and only 0.32 for shared input repetitions. The results show that VoIP user recognition by key-click patterns is possible. RF algorithm gives better performance than the other two. It is easier to recognize users if they have different PIN codes to input.

4.5 Discussion on Countermeasure

One countermeasure is to insert random delay between key-click events. When a user-agent receives a key-click event, it does not immediately process the event. Instead, it puts all information of the event (e.g., the holding time) in a First In First Out (FIFO) queue. The user-agent constantly checks the queue and fetches a event from it after a random time delay for processing. It obscures real key-click patterns. Yet another defending method is to encrypt the whole RTP packet using IPSec. We know that the attacks take advantage of the factor that SRTP does not encrypt RTP headers, which enable attackers to restore key-click patterns. The attacks do not work if the RTP headers are encrypted. RFC 3711 [10] suggests IPSec (ESP method) [19] if users would like both the RTP headers and contents to be protected. Nevertheless, users may particularly worry about the performance overhead and configuration complexity by using IPSec. Although it is possible to effectively reduce the performance overhead by using packet header compression [9], configuration complexity might be a barrier to widely deploy IPSec in VoIP.

5 Related Work

Keystroke dynamics is a method to recognize individual users by using their typing characteristics, with the time stamps of key-down and key-up. Many

previous work has been done in this domain and a broad overview can be found in [23]. This section only summarizes the work most relevant in the context of ours. Maxion et al. [22] asked 28 volunteers to type 200 repetitions of the same 10-digital code using only the index finger on the number pad of a standard keyboard. They intercepted the keystroke information on key-down and key-up time locally and analyze them using random forest classifier. Half of the data were selected for training and the rest were used for testing. The classifier achieved a hitting rate of 99.54% and false alarm rate of 12.50%. Kotani, et al., [21] performed experiments using a special pressure sensitive keypad with 9 subjects. Each subject typed 20 repetitions of the same 4-digital PIN code. Besides key-down and key-up times, stroking force was chosen as a third element. Their classifier gives a equal error rate at 2.4%. Clarke et al., [17,15,16] performed several tests in which they asked different number of subjects (from 16 to 32) to type the same 4-digital PIN code for 30 repetitions on the keypad of a mobile phone. 20 repetitions are used for training and the rest is used for testing. Their neural network classifier gives an equal error rate from 5.5% to 8.5%. Our work is on a different scenario. We recover key-click pattern from VoIP flows and the subjects use a mouse and virtual keypad for input. Moreover, our method needs to take the networking condition impact into account.

On VoIP user recognition, Khan et al. [20] proposed a scheme exploiting patterns on the sizes distribution of RTP voice packets. Their classifier achieved a hitting rate of 75% for 10 speakers and 51% for 20 speakers. Similarly, Backes et al. [8] proposed an approach using the periods of speech and silence of a speaker in a conversation. This speaker specific pattern is modeled using the speaker's talking speed and frequency. The identification rates obtained were 65% for 13 speakers and 48% for 20 speakers. Nevertheless, their method requires specific codec being applied. Our work achieves the same goal but in another way: We exploit the side channels in RTP event flows rather than RTP voice flows.

6 Conclusion

This paper proposed an attacking method to recognize VoIP users for user-level calling records profiling. It takes advantage of user-specific key-click patterns. Even if a VoIP flow are protected by SRTP, attackers can still recover key-click patterns from the flow by reading packet header fields. The impact introduced by varying network conditions (jitter, packet loss) can be minimized. In an empirical setup with 31 users our analysis is able to correctly classify unknown RTP flows in about 65%. For user verification, the average equal error rate is from 10% to 29%. The result raises serious concerns about anonymity for VoIP users. To prevent this attack, users can consider to either randomize the time interval between key-clicks or use another security scheme (e.g., IPSec) which not only encrypts the whole part of a RTP packet, but also pads all RTP packets to an equal size.

There are still some limitations on our currant experiments. First, we only asked subjects to enter PINs using the virtual keypad by mouse clicking so far.

Nevertheless, some user-agents also support standard keyboard input. Second, we let all subjects to enter the same PIN "9913" for collecting shared input repetitions. The results with only one PIN instance may difficult to be generalized. In future work, we will investigate this problem further not only using virtual keypad, but also standard keyboard. In addition, we will try several shared PIN candidates for collecting shared input repetitions.

References

1. 40 websites offering telephone calling records and other confidential information, http://epic.org/privacy/iei/attachment_a.pdf (visited at November 15, 2010)
2. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien, http://cran.r-project.org/web/packages/e1071/index.html (visited at September 18, 2010)
3. randomForest: Breiman and Cutler's random forests for classification and regression, http://cran.r-project.org/web/packages/randomForest/ (visited at September 18, 2010)
4. rpart: Recursive Partitioning, http://cran.r-project.org/web/packages/rpart/ (visited at September 18, 2010)
5. TCPDump, http://www.tcpdump.org/ (visited at July 20, 2010)
6. The R project for statistical computing, http://www.r-project.org/ (visited at July 18, 2010)
7. X-Lite, http://www.counterpath.com/x-lite.html (visited at July 18, 2010)
8. Backes, M., Doychev, G., Dürmuth, M., Köpf, B.: Speaker Recognition in Encrypted Voice Streams. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 508–523. Springer, Heidelberg (2010)
9. Barbieri, R., Bruschi, D., Rosti, E.: Voice over ipsec: Analysis and solutions. In: Proceedings of ACSAC 2002. IEEE, Los Alamitos (2002)
10. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The Secure Real-time Transport Protocol (SRTP), RFC 3711 (2004)
11. Bennett, K.P., Campbell, C.: Support vector machines: hype or hallelujah? SIGKDD Explor. Newsl. 2(2), 1–13 (2000)
12. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
13. Breiman, L., Stone, C.J., Friedman, J., Olshen, R.A.: Classification and Regression Trees. Chapman & Hall/CRC, Boca Raton (1984)
14. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of ICML 2006. ACM, New York (2006)
15. Clarke, N., Furnell, S.: Advanced user authentication for mobile devices. Computer & Security 26, 109–119 (2007)
16. Clarke, N., Furnell, S.: Authenticating mobile phone users using keystroke analysis. International Journal of Information Security 6, 1–14 (2007)
17. Clarke, N., Furnell, S., Lines, B., Reynolds, P.: Using keystroke analysis as a mechanism for subscriber authentication on mobile handsets. In: Proceedings of SEC 2003. Kluwer, Dordrecht (2010)
18. Egevang, K., Francis, P.: The IP Network Address Translator (NAT), RFC 1631 (2006)
19. Kent, S., Seo, K.: Security Architecture for the Internet Protocol, RFC 4301 (2005)
20. Khan, L.A., Baig, M.S., Youssef, A.M.: Speaker Recognition from Encrypted VoIP Communications. Digital Investigationg (2009)

21. Kotani, K., Horii, K.: Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics. Behaviour & IT 24(4), 289–302 (2005)
22. Maxion, R.A., Killourhy, K.S.: Keystroke biometrics with number-pad input. In: Proceedings of DSN 2010. IEEE, Los Alamitos (2010)
23. Peacock, A., Ke, X., Wilkerson, M.: Typing patterns: A key to user identification. IEEE Security and Privacy 2(5), 40–47 (2004)
24. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A transport protocol for real-time applications, RFC 3550 (2003)
25. Schulzrinne, H., Taylor, T.: RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals, RFC 4733 (2006)
26. Sengar, H., Ren, Z., Wang, H., Wijesekera, D., Jajodia, S.: Tracking skype voip calls over the internet. In: Proceedings of INFOCOM 2010. IEEE, Los Alamitos (2010)
27. Wang, X., Chen, S., Jajodia, S.: Tracking anonymous peer-to-peer VoIP calls on the Internet. In: Proceedings of CCS 2005. ACM, New York (2005)
28. Wu, C., Chen, K., Chang, Y., Lei, C.: Speaker Recognition in Encrypted Voice Streams. In: Schulzrinne, H., State, R., Niccolini, S. (eds.) IPTComm 2008. LNCS, vol. 5310. Springer, Heidelberg (2008)

Problem Analysis of Traditional IT-Security Risk Assessment Methods – An Experience Report from the Insurance and Auditing Domain

Stefan Taubenberger[1], Jan Jürjens[2], Yijun Yu[3], and Bashar Nuseibeh[3, 4]

[1] MunichRe, Munich, Germany
Staubenberger@munichre.com
[2] TU Dortmund and Fraunhofer ISST, Germany
http://www.jurjens.de/jan
[3] Lero, University of Limerick, Ireland
y.yu@open.ac.uk
[4] The Open University, Milton Keynes, United Kingdom
b.nuseibeh@open.ac.uk

Abstract. Traditional information technology (IT) security risk assessment approaches are based on an analysis of events, probabilities and impacts. In practice, security experts often find it difficult to determine IT risks reliably with precision. In this paper, we review the risk determination steps of traditional risk assessment approaches and report on our experience of using such approaches. Our experience is based on performing IT audits and IT business insurance cover assessments within a reinsurance company. The paper concludes with a summary of issues concerning traditional approaches that are related to the identification and evaluation of events, probabilities and impacts. We also conclude that there is a need to develop alternative approaches, and suggest a security requirements-based risk assessment approach without events and probabilities.

Keywords: IT risk analysis, IT risk assessment, Security requirements.

1 Introduction

Companies and governmental organizations are interested in detecting and mitigating the risks of possible profit and image losses. Many quantitative and qualitative methods and toolkits for Information Technology (IT) security risk analysis have been developed using, such as normal probability, Bayesian probability, Fuzzy theories, Annual Loss Expectancy (ALE), all of which are based on probabilities and events as the risk is "measured in terms of a combination of the likelihood of an event and its consequence" in the ISO 27005 standard [19]. Estimating risks reliably with precision is difficult because of their unpredictability according to this definition: in each traditional risk assessment method or toolkit, probabilities about the events and the possible consequences have to be determined, and each of the steps to determine risk – identifying events, determining probabilities and impacts – has weaknesses, making risk assessments prone to errors, unreliable, and results questionable.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 259–270, 2011.

The objective of this paper is to discuss the general issues of determining risk with events and probabilities within traditional approaches. We report on our experiences in IT risk assessments in the insurance and auditing domain. The main contribution of the paper is the thorough analysis of the problems of traditional risk assessment approaches based on the literature and our experiences. The paper is structured as follows: in section 2 we present problems of traditional approaches related to underlying methods and risk assessment steps. In section 3 we report on our problem experiences applying a traditional approach by hand on a real world example. We summarize the issues of traditional risk assessment approaches in section 4 and in section 5 we suggest developing alternative risk assessment approaches, such as based on security requirements and business process models.

2 Traditional Approaches to IT-Security Risk Assessment

In the literature many approaches for IT security risk assessment are available to researchers and practitioners. Discussions of available approaches can be found in Ralston et al. [23], Alter and Sherer [3], ENISA [7], and Putnam [21]. We discuss these methods from the perspective of the used underlying assessment methods (qualitative or quantitative), risk assessment activities, and the selection of assessment methods. In the following, we present our critiques based on a literature review.

2.1 Qualitative and Quantitative Methods

Quantitative risk assessment methods use numeric probability where the probability expresses the knowledge that the event occurs. With quantitative approaches risk is determined by the probability of an event and the likelihood of a loss. Examples of use of quantitative methods are: normal probability, Bayesian probability, Fuzzy theories and Dempster Shafer theory, Monte Carlo Simulation [15], Annual loss expectancy (ALE), and stochastic dominance [22]. The advantages of quantitative methods are that IT assets are identified most likely for damages [22], measures can be used for the impact magnitude and be directly compared, [8]. The disadvantages of quantitative methods are that there are no exact probability values of loss at the time when they are estimated and half of the estimates are statistically either too high or too low [22]. Furthermore, the probability function that usually follows a normal distribution may be deformed because it represents average values of a few extremes and many low ones [22]. Additionally, a scale has to be provided for what the value of “x” percent means. These values have to be translated to a literal meaning.

Qualitative risk assessment methods use non-numeric values or number ranges to express the risk as descriptive values [22]. Examples for qualitative methods are: scenario analysis, fuzzy metrics, questionnaires [22], preliminary risk analysis (PHA), hazard and operability study (HAZOPS), and failure mode and effects analysis (FMEA/FMECA) [24]. The advantages of qualitative methods are that these approaches are time and cost efficient because no exact value has to be determined and they are valuable in estimating risk approximately [22] as well as areas of improvement can be easily identified. However, the disadvantage of qualitative methods is

that they are not precise as the value is expressed within a spectrum that has to be understood by all involved parties [22]. Additionally, methods provide no measurement for the impact and therefore it is difficult to conduct a cost benefit analysis [30]. Although quantitative and qualitative methods can be combined and used together [17], results combination and interpretation become more difficult because different rating scales, underlying assessment principles or the variances in risk weighting are difficult to mix up.

2.2 Risk Determination

In internationally accepted standards or methods like Octave [2], CORAS [29], AS/NZS 4360 [6] or ISO 27000 standards, the principal steps to determine risks are asset identification, event/threat identification, vulnerability/control identification, likelihood determination and impact analysis. Within the literature many issues of or critique on traditional approaches are provided. We can categorise all of these into three areas: identification, data and assessment.

(1) The *identification* category is about activities to determine, e.g. an event. A threat that uses vulnerabilities is defined as an event [19]. The identification of threats and vulnerabilities is challenging as underlying conditions change constantly e.g. development of new technologies, new competitors, new laws, etc. [16], [14] Therefore, threats and vulnerabilities are not static, with their behaviour and seriousness change within days. Threats and vulnerabilities are identified based on security expert knowledge, usage of security scanning tools and public available data. Security experts use implicit knowledge and experiences as well as explicit data such as vulnerability lists for the risk identification. But how do we know and how can we verify whether or not all threats and vulnerabilities have been identified correctly and completely? Furthermore, events in associated companies (e.g. outsourcing partners or inter-company process chain partners) could not be discovered as there are beyond company boundaries. However, these events could negatively affect a company as business processes and systems are heavily interconnected nowadays [16].

(2) The *data* category is about data needed for the evaluation of risks. For the impact and probability assessment of a risk, data regarding the impact and probability of event in a given situation for systems is needed. The major issues here are that exhaustive public available data of occurred events, impacts and their probabilities are not available [27], and the internal historic data are not available for the estimation of possible change impacts on the company. For example, the event has not occurred in this type of industry yet, within the company or the scope in this situation. If no comparable data is available, best guesses must be used for determining the change impact and probability. But how to make such a best guess in an environment where we do know little about the basic population to determine the occurrence rates, effects or the change impacts of the events? In case that event data is available, internal data about events in companies can still be incomplete or may represent a “lucky” history [9] and get quickly obsolete. In addition, internal historic event data may not represent a true view and events recorded could be lower-than-average [9]. For example, the claims data recorded regarding the occurrence rate and extent of loss is often below the average of the reference industry or competitors. Another issue is that probability

distributions get incorrect as they are based on historic data not representing event behaviour changes [28]. For example, the 100-year events reoccur nowadays for every 10 years in fat-tailed distributions. How can we ensure or verify that the data used for the assessment based on such data is still correct?

(3) The *assessment* category is about activities or models to evaluate the change impacts. Risk assessment is based on the impact and the probability of the event. The models used to determine risks and dependencies are poor because co-occurrence of risks, uncertainty between event relations and different assessment scales are not considered. Co-occurrence of events within different or the same risk leads to indeterminable impacts and damages because the events might occur in associated companies that may have an impact on other risks that are not considered when they are evaluated on their own. In current methods, the assessments are performed on decomposed model elements but do not consider the organization as a whole. Furthermore, there is uncertainty between the relation of an event and the impact by its nature. For example, the impact of the event is not known or dependent on other conditions/parameters. However, side effects (multiple impacts or dependencies) or parameters are not considered and uncertainty is assessed by gut feelings or subjective security expert knowledge [27]. Although safeguards put in place are considered in the impact assessment, they are evaluated for a particular threat/vulnerability, and the side effects of other events are not considered. How do we determine that safeguards are operated as intended? A systematic assessment of the safeguards regarding secure operation, secure design and effectiveness is currently missing. Furthermore, probabilities are measured by different techniques; for example, by quantitative and qualitative methods. But the comparability of qualitative and quantitative assessments of risks or probabilities within an assessment method is not validated. Furthermore, assessments are influenced by perceptions. Behavioural biases outgoing of the educational background, organizational level or positive/negative attitude of the assessor may affect the assessment of events, probabilities of occurrence or the impact estimation [16],[27],[25]. In addition, current risk assessment proceedings lead to simplification and are focused to strong on technical issues rather than on information or business issues [11]. For procedural reasons the assessor will usually simplify otherwise he will be lost in detail and forget the objectives [12]. Additionally, methods follow the waterfall model and therefore are not capable of considering changes during the lifetime of the assessment [31].

2.3 Selection and Classification of IT-Security Risk Assessment Methods

In the literature many approaches for IT security risk assessment are available by researchers and practitioners and in general “published work related to risk assessment is very difficult to categorize.” ([23], p.6) and “There are more than 200 risk management methods making it a challenge to select the most adequate one” ([18], p.1). These difficulties to categorize and select an appropriate risk assessment approach arise because the risk assessment process consists of different phases namely: risk identification, risk analysis, risk assessment (evaluation and ranking) and risk management (treatment and mitigation), and developed approaches cover different phases as well as concentrate on different aspects, problems or business areas. An issue in classifying approaches is to determine how much of the risk assessment

process is covered by the proposed approach. Another issue is the great variety and profundity of the approaches and their description how they perform and to apply in a given situation. Researches tried to classify approaches like Campell and Stamp [5] who provided a classification scheme consisting of two dimensions "level" and "approach" divided further into subcategories, however lacks a classification regarding the elements of the risk assessment approach. The five basic classes used by Siponen [26] misses any further distinguishing characteristics and are therefore not expedient for a classification. In an ENISA working group paper [1] as well as in the thesis of Poettinger [20], risk exposure, risk impact and impact segment are used to determine the most appropriate risk assessment methodology. Although Spider diagrams are used to compare the methods with organizational requirements, currently there is no general accepted and proven classification scheme in existing approaches. Further on, developed or criticized approaches are typically not classified or categorized making it hard for researches to apply the approach in the correct setting or to select the most appropriate one.

3 Our Experiences with Traditional Approaches

In Information System (IS) audits as well as for providing insurance cover for business interruptions auditors have to evaluate IT risks. They evaluate IT systems, processes and risk prevention capabilities. The purpose of these risk assessments is to determine significant risks that are associated with the design, implementation and operation of IT systems of a company. The audit committee or the insurer commissions these assessments. The audit team presents these significant risks to the management or underwriters and reports to the audit committee or insurer. The significance of risks is determined qualitative by the impact of the threat and the results are used to decide about risk acceptance/ mitigation or about insurance cover. However, auditors face the problem that data about events, probabilities are rarely available in public or in the company assessed. Furthermore, these audits have to be cost and time efficient and the results should be reliable regarding future events to acquire profitable business as well as for the annual financial statement.

3.1 IT-Security Risk Assessment with a Traditional Approach

In this section we describe the context and the results of applying a traditional approach on a simplified real world example.

Context: We have applied a proven traditional approach such as [30] more than ten times to determine IT security risks in subsidiaries and branches of a reinsurance company within audits as well as at companies that applied for business interruption insurance. These assessments conducted by an IT auditor and an independent security expert, focus on IT management, systems, and normally last one week. The assessment team is independent of the IT operation or IT management of the assessed company or branch. IT departments of different sizes and organizational forms were assessed. In a centralized environment the assessments stopped at the service interface; however service quality and service agreements were considered.

Approach selection: We selected the NIST 800-30 approach [30] because it is well known and documented, learnt in less than three days [4] and the tendency to rate threats as medium or low [4]. Especially, the tendency to have a few high risks is important to direct management/companies efforts to the most critical issues.

Limitations: With risk assessments not all risks may be identified neither can we guarantee that. However, we adjusted our assessments to identify significant risks regarding best practices within a confidence level of professional experience.

The following is a simplified real world example: The main sales channel of an internet retailer for clothes is their online web store. The customer has to provide all shipping and payment data before an order via the online store is processed. Customers can make payments by credit card or on delivery. After providing and verification of all necessary data, the order is stored and processed. The web store is a web application with a connected database containing all order data and has an interface to a third party service to verify credit card data. As we have thorough knowledge about vulnerabilities, we know that the web application has an SQL injection problem and an encryption problem in the communication with the costumer and third party service.

An IT security risk assessment with a traditional approach such as [30] would proceed with asset- , threat-, vulnerability- and impact-analysis to determine risks.

(1) Asset identification and analysis: Hardware, software, data, people have to be identified as well as their criticality or value to the organization. In our example we identified the online web store, the external service provider, customer, credit card and order data. People involved include customers and order handling personnel.
(2) Threat identification and analysis*:* All potential threat sources have to be identified. We identified natural disaster threats such as tornados, floods and earthquakes, and human behaviour threats from hackers, computer criminals, terrorists or espionage and insiders/disgruntled employees. Technical threats include blackouts, fire, and chemical pollution.
(3) Vulnerability identification and analysis: All weaknesses that can result in security breaches in the system security procedures, design or operation have to be determined. A system design analysis revealed that the web server application has an SQL injection problem and that the communication with the database is unencrypted. The external service provider was not analysed as an external report showed no vulnerabilities. Employees were not considered as vulnerable as there is no customer contact and no indications of disgruntled employees.
(4) Likelihood determination and impact analysis: The impact and the likelihood of a successful security breach have to be determined with regard to the criticality of the asset. The probability ratings were defined as low (0-30%); medium (30-70%) and high (70-100%). The impact scale was defined as low (<1 million Euro), medium (1 to 5 million Euro) and high (>5 million Euro).

Natural disaster threats were not considered because the data centre is not exposed and estimated probabilities are < 1 percent. The power blackout from the technical threats was rated as probable (low) but with low impact. Fire is no risk as it is treated by a sprinkler system. Chemical pollution was rated as unlikely. The web server encryption issue was rated with low probability for criminals, medium for hackers and the impact was rated medium for both. The web server injection issue was rated with

low probability and medium impact. Terrorists and espionage was not considered because the business is not critical. Table 1 shows some of the risk ratings.

Table 1. Risks and risk ratings in our scenario

Traditional approach		
Risk	**Probability**	**Impact**
Power Blackout	Low	Low
Web server encryption criminal	Low	Medium
Web server encryption hacker	Medium	Medium
Web server SQL injection	Low	Medium

3.2 Methodological and Estimation Problems

In the following we describe general and probability estimation issues we experienced by applying the steps of the NIST 800-30 [30] approach. In the ***asset identification phase***, the business process is decomposed into single elements. But any dependencies between elements are neither considered nor modelled. In the ***threat and vulnerability identification phases*** the main problem is uncertainty. We do not know whether the threats listed or the identified vulnerabilities are complete and comprehensive and how to verify them. We are dependent on publicly available data and the assessor knowledge and experience. For the ***likelihood determination and the impact analysis of threats*** there is no detailed guidance available. For example, NIST 800-30 does not describe how to link threat sources with vulnerabilities and how to derive or evaluate any probabilities. Our probability estimates may not represent a true view as the behaviour of attackers and defenders changes. The aggregation of probability values causes further problems as the probability of occurrence might be misrepresented. In addition, the consequences and the existence of misestimating are not considered. Misestimating or unknowingly influenced assessors [27] as well as the existence of ambiguity and the aggregation of risk creates an estimation risk that is not considered. As a result, events and impacts may be under-/over-represented.

In what follows we demonstrate the divergence of probability estimates. Therefore, we try to verify our probability ratings of section 3.1. We attempt to determine the probability that a malicious user exploits the encryption weakness of the web server and the probability not exploiting any weakness. For determining these probabilities the following parameters should be considered:

- Number of known exploits and not secured exploits for the web server version: Determinable by publicly reported bugs/vulnerabilities and a security analysis.
- Criticality of exploits: Determinable as exploits are rated.
- Detection rate of all vulnerabilities by malicious user: Not determinable as the ratio is dependent on the knowledge of vulnerabilities, the used/ available tools and number of exploits/ vulnerabilities available.
- Number of users: Determinable by page views and IP-address matching.
- Ratio of successful exploiting: Not determinable as the ratio is dependent on malicious user's knowledge, the complexity of vulnerabilities as well as the motive, resources and time of the malicious user.

- Relation of friendly and malicious users accessing the web server: Not determinable and dependent on e.g. popularity of the company, monetary gain.
- Impact of controls: Is implicitly considered in the successful exploiting ratio.

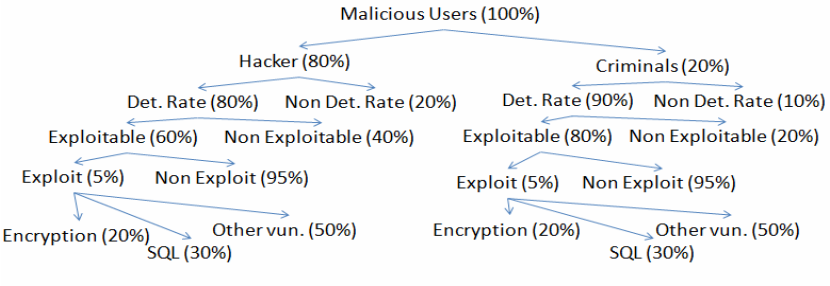


Fig. 1. Dependency tree with probability ratings

A parameter tree showing dependencies and assigned probabilities values for our example in section 3.1 looks like figure 1. The percentages in the probability tree were assigned by us based on available data and estimates. The probabilities for events as asked in the beginning are as follows, if one computes the probabilities down the probability tree for a hacker or a criminal.

- A criminal exploits the encryption weakness in the web server in 0.144%.
- A hacker exploits the encryption weakness in the web server in 0.384%.
- A malicious user exploits the encryption weakness in the web server in 0.528%.
- The likelihood that a malicious user does not exploit any weakness is 97.36%.

The values express the probability of occurrence of exploiting the vulnerability by a malicious user. Notice, that there is a major discrepancy between the results of section 3.1 and this calculation. These result variations maybe caused by us because of bad estimates or mistakes. Therefore, we also tried changing ratios besides the variations while we recognized the following:

Dependencies: There is a direct dependency of the result to single parameters e.g. a reduction/increase of one parameter from 5 to 10 (100 percent change) leads to a reduction/increase of the result in the same percentage. We recognized that the percentage of misestimating is relevant not the absolute amount.

Baseline: The total population has to be specified because a, for example, 12% or medium probability has no significance. This is especially important when populations are linked like the malicious users to normal user's ratio.

Probability: In a chain of parameters the total probability inclines against 0 or 100 percent as it is below or above the minimum or maximum values. These high or low values blur the total probability exceptionally.

Tree diagram: Generally, it is difficult to determine the dependencies of parameters, the correct tree diagram and to verify the diagram as there is no data available.

Perception: The perception of the results is dependent on the probability question and the result value. A higher percentage and positive statement (e.g. an event is 80 percent likely instead of 20 percent unlikely) is assumed to provide more confidence.

3.3 Result Presentation and Perception

Traditional approaches present risks as threats or threat diagrams. Categories such as high, medium and low indicate the severity and probability of the threat like shown in table 1. However, without further information, like basic population, countermeasures costs, required security, and effects on operations and the security of the application, data or transaction, a reasonable decision on risk mitigation or acceptance is hardly possible. Furthermore, the decision on risk mitigation or acceptance is a second assessment influenced by subjective factors and on individual's perception of risk [27] representing constraints to countermeasure implementation.

Risk attitude and perception: The perception of risk is influenced by e.g. personal experiences, media, social groups [27] as well as a person's risk attitude - risk taker vs. risk aware person.

Frequency: Countermeasure implementation is dependent on costs, impact, probability and frequency. But the frequency in a period of time is not specified in the risk analysis results.

Cost objectives: The implementation of measures depends on company internal cost objectives as personal or departmental objectives may not be accomplished. Furthermore, measures that are not planned in the current year budget may not be implemented immediately.

Prioritization: Business critical projects or security issues in daily operations have a higher priority than proposed countermeasures as an event has materialized.

All these factors are influencing the overall assessment results and arise due to the representation of risk and the possibility to interpret results. As a result the optimal security level is not achieved and the company's security standard was defined, changed (without notification) or violated by the acceptance of risk.

4 Problem Analysis - Summary

In the literature review as well as through our experiences we have identified a number of issues related to used methods and activities in traditional risk assessments.

Methods: Quantitative methods base on data that is not reliable available in practice with a certain precision and qualitative methods provide results within a range with deviations that have to be interpreted. Interpretation is subject to misjudgment and the selection of an approach in a given situation is not supported by any of the developed methods making it difficult to choose the appropriate approach.

Guidance and identification: Current standards are missing guidance on likelihood determination, event correlation and linking of threats, probabilities and impact. However, such guidance would be highly beneficial for risk analysts. The concept used for identification of, threats, vulnerabilities or correlations "you know one when

you see one" applied in most methods does not work on new risks as this concept is based on implicit experience, thresholds and occurred damages.

Dependencies: In current approaches, the assessor conducts the risk assessment on decomposed single model elements. However, that proceeding neglects design or interrelated risks as well as organizational coherences.

Probabilities: The assessor mostly estimates probability values in assessments, as there is no reliable and true data. But estimates are mostly biased, statistically incorrect or the data base of the distribution might be incorrect because of behavioural changes or a "lucky" history. In addition, we experienced that already small derivations or unconsidered parameters such as the timeframe or total population have a material impact on the result. There is no feasible way to verify the correctness and completeness of probability dependencies (tree) and positive and high probability statements are perceived as more trustworthy by people.

Assessment: Assessments are conducted on uncertainty regarding events, probabilities and impact. However, uncertainty, co-occurrence as well as dependencies are not modelled and properly considered. Furthermore, assessments are specific to a point of time not considering environment changes or prevention capabilities of the company.

Risks results: We experienced that low and a few medium risks were not mitigated because of personal and company specific constraints such as perception, cost objectives and prioritization of activities. Furthermore, the impact on the companies security level or polices is not appropriately considered when risk is accepted.

Environment: To identify and to determine events, probabilities and impacts correctly we must have comprehensive knowledge about the environment of the risk, the company and outside world. This would require that all parameters, corresponding probabilities, the basic population as well as correlations are known, are immediately updated, base on enough statistic data and could be modelled. But comprehensive knowledge about the environment is not available, may be compromised, cannot be verified and cannot be modelled as the real world is too complex and unpredictable. This applies to all risk assessments and is not specific to our problem domain. Furthermore, risk is about people. Their behaviour is not objective or rational, may follow personal interests or herd-instincts and be biased.

5 Conclusion

Due to the nature of risk - its unpredictability and complexity - risk assessment is difficult. Our problem analysis of traditional approaches based on the literature and our experiences in the insurance and auditing domain showed that such issues like uncertainty, wrong estimation and perception are mainly associated with determining events, probabilities and change impacts, affecting adversely the risk results. We therefore suggest that future approaches should attempt to determine risk by alternative concepts.

One possible and promising direction is to use security requirements (SR) [10], [13] not only for determining the impact of a threat or the seriousness of vulnerabilities but considering organizational needs in the risk assessment. An alternative risk assessment approach with SR could manage to determine risk without using events and

probabilities and considering the organizations capability to handle and prevent events. This could be achieved, for example, by specifying the business process data security needs and by evaluating these requirements by hand of process model activities concerning the actors related to the system. Security requirements and corresponding security controls are evaluated at individual process activities for validating whether or not the system implementation, the actor's process activities (operation) and the process design adheres to the requirements. In addition to business process evaluation, IT process maturity and performance are evaluated to detect weaknesses, to determine operating effectiveness and prevention capabilities. The IT process evaluation results can be used to evaluate the adherence of business process data security requirements from an infrastructure perspective as well as to indicate the time invariance of the risk results. However, to determine risks only using security requirements without having to determine events and probabilities would lead to a redefinition of risk as "the non-adherence of security requirements thereby causing harms to the organization regardless of a point in time". An advantage of such an approach would be that assessment results are more time independent and results probably more accurate and linked to organizational security needs. Before developing such an approach we believe we need a better understanding of the interaction of security requirements, risk treatments, risks, assets and assurance as a foundation. We are confident that a security requirement based approach has the potential to overcome the limitations of traditional approaches and we hypothesise that an entity would face no substantial risks from any events/threats if the evaluated security requirements have been adhered to.

Acknowledgement. Supported, in part, by the EU as part of the SecureChange project and SFI grant 03/CE2/I303_1.

References

[1] ENISA 2007-2008 ad hoc Working Group on Risk Assessment/Risk Management. Determining your organization's information risk assessment and management requirements and selecting appropriate methodologies (2008)
[2] Alberts, C., Dorofee, A., Stevens, J., Woody, C.: Introduction to the OCTAVE Approach. Carnegie Mellon Software Engineering Institute, Pittsburgh, USA (August 2003)
[3] Alter, S., Sherer, S.: A general, but readily adaptable model of information system risk. Communications of the Association for Information Systems 14, 1–28 (2004)
[4] Buyens, K., DeWin, B., Joosen, W.: Empirical and statistical analysis of risk analysis-driven techniques for threat management. IEEE Computer Society, Los Alamitos (2007)
[5] Campbell, P., Stamp, J.: A classification scheme for risk assessment methods. Sandia Report, Sand2004-4233 (2004)
[6] Australian/New Zealand Standards Comittee. Risk management ASNZ 4360:1999 (1999)
[7] ENISA. Inventory of risk assessment and risk management methods, ENISA ad hoc working group on risk assessment and risk management (March 2006)
[8] Feather, M., Cornford, S.: Relating risk and reliability predictions to design and development choices. In: Proceedings of the Annual Reliability and Maintainability Symposium (RAMS), Newport Beach, CA, January 23-26 (2006)
[9] Frachot, A., Roncalli, T.: Mixing internal and external data for managing operational risk (2002)
[10] Gerber, M., von Solms, R.: From risk analysis to security requirements. Computers & Security 20, 577–584 (2002)

[11] Gerber, M., von Solms, R., Overbeek, P.: Formalizing information security requirements. Information Management & Computer Security 9(1), 32–37 (2001)
[12] Halliday, S., Badenhorst, K., von Solms, R.: A business approach to effective information technology risk analysis and management. Information Management &Computer Security 4(1), 19–31 (1996)
[13] Houmb, S., Jürjens, J.: Developing secure networked web-based systems using model-based risk assessment and UMLsec. In: 10th Asia-Pacific Software Engineering Conference (APSEC 2003), Chiangmai, Thailand, December 10-12 (2003)
[14] Jackson, M.: NII-OU Security Workshop @ The Open University (November 2007)
[15] Kaplan, S.: The words of risk analysis. Risk Analysis 17(4) (1997)
[16] Kinney, W.: Research opportunities in internal auditing - chapter 5 auditing risk assessment and risk management process. The Institute of Internal Auditors Research Foundation (2003)
[17] Zhang, Y., Jiang, S., Cui, Y., Zhang, B., Xia, H.: A qualitative and quantitative risk assessment method in software security. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 1, pp. V1-534–V1-539 (2010)
[18] Matulevius, R., Mayer, N., Mouratidis, H., Dubois, E., Heymans, P., Genon, N.: Adapting Secure Tropos for Security Risk Management in the Early Phases of Information Systems Development, pp. 541–555. Springer Publishing, Heidelberg (2008)
[19] International Organization of Standardization (ISO). ISO 27005 Information technology - Security techniques - Information security risk management, International Organization of Standardization (ISO) (2008)
[20] Pöttinger, J.: Self assessed risk management. Master's thesis, Fachhochschul-Masterstudiengang Sichere Informationssysteme (2009)
[21] Information Security Management References, Corporate Information Security Working Group, Chairman: A. Putnam, Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census, Government Reform Committee, United States House of Representatives, Mapping of Existing Work on Infosec (Best Practices) Subgroup: C. Kreitner, M. Rasmussen, Coordinators (2004)
[22] Rainer, R., Snyder, C., Carr, H.: Risk analysis for information technology. Journal of Management Information Systems 8(1), 129–147 (1991)
[23] Ralston, P., Graham, J., Patel, S.: Literature review of security and risk assessment of SCADA and DCS systems, Technical Report TR-ISRL-06-01 (July 2006)
[24] Rausand, M.: Risk Analysis An Introduction. In: System Reliability Theory, 2nd edn. Wiley, Chichester (2004)
[25] Redmill, F.: Risk analysis - a subjective process. Engineering Management Journal 12(2), 91–96 (2002)
[26] Siponen, M.: An analysis of the traditional is security approaches: implications for research and practice. European Journal of Information Systems 14, 303–315 (2005)
[27] Stewart, A.: On risk: perception and direction. Computers & Security 23, 362–370 (2004)
[28] Stiglitz, J.: Making globalization work: Global financial markets in an era of turbulence. Frankfurt (February 2008)
[29] Stølen, K., den Braber, F., Dimitrakos, T., Fredriksen, R., Gran, B.A., Houmb, S., Lund, M., Stamatiou, Y., Aagedal, J.: Model-based risk assessment – the CORAS approach. In: NIK Informatics Conference 2002, Kongsberg (2002)
[30] Stoneburner, G., Goguen, A., Feringa, A.: NIST Special Publication 800-30: Risk Management Guide for Information Technology Systems. National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899-8930 (July 2002)
[31] Vidalis, S.: A critical discussion of risk and threat analysis methods and methodologies. Technical Report CS-04-03, University of Glamorgan, Pontypridd (2004)

On Computing Enterprise IT Risk Metrics

Sandeep Bhatt, William Horne, and Prasad Rao

Cloud and Security Lab
HP Laboratories
5 Vaughn Drive, Princeton NJ 08540, USA
{sandeep.bhatt,william.horne,prasad.rao}@hp.com

Abstract. Assessing the vulnerability of large heterogeneous systems is crucial to IT operational decisions such as prioritizing the deployment of security patches and enhanced monitoring. These assessments are based on various criteria, including (i) the NIST National Vulnerability Database which reports tens of thousands of vulnerabilities on individual components, with several thousand added every year, and (ii) the specifics of the enterprise IT infrastructure which includes many components.

Defining and computing appropriate vulnerability metrics to support decision making remains a challenge. Currently, several IT organizations make use of the CVSS metrics that score vulnerabilities on individual components. CVSS does allow for environmental metrics, which are meant to capture the connectivity among the components; unfortunately, within Section 2.3 of [1] there are no guidelines for how these should be defined and, consequently, environmental metrics are rarely defined and used.

We present a systematic approach to quantify and automatically compute the risk profile of an enterprise from information about individual vulnerabilities contained in CVSS scores. The metric we propose can be used as the CVSS environmental score. Our metric can be applied to the problem of prioritizing patches, customized to the connectivity of an enterprise. It can also be used to prioritize vulnerable components for purposes of enhanced monitoring.

1 Introduction

Deciding which subsystems within a large enterprise system should be prioritized for patching or should be monitored more closely is an art. Patching involves more than the deployment of a patch, which is itself a non-trivial task. It also involves assessing the risks of a vulnerability; planning, scheduling, testing and qualifying the patches in a staging environment; and finally assembling the resources needed for deployment, and for handling patch distribution failures and help desk calls from end users. It is estimated that every patching event costs anywhere between 0.0025 and 0.5513 person hours per system [3]. For large organizations with many tens of thousands of systems to be managed, the cost of security patch management can be excessive.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 271–280, 2011.

As a result, organizations cannot address every known vulnerability, but rather must prioritize them, based on the risk they pose to the enterprise. How should an IT organization assess the risk posed by component vulnerabilities? The key concern is to design a rational risk assessment scheme that can be automated.

Several IT organizations, prioritize security patches using the Common Vulnerabilities and Exposure (CVE) system. CVE is a system for disseminating vulnerability information against various types of IT assets. CVE reports are coupled with a set of metrics defined by the Common Vulnerability Scoring System (CVSS). CVSS includes a base score, which rates the vulnerability in isolation. CVSS also specifies environmental metrics intended to allow organizations to account for the relevance of the vulnerability in their environment. Unfortunately within Section 2.3 of [1], there are no guidelines to develop environmental metrics and so they are rarely, if ever, used.

The risk presented by a component vulnerability depends on the context in which the component is used, as well as the location of the component within the enterprise topology. A component with a severe vulnerability in an isolated part of a network that has little relevance to business operations may not pose as much risk as a component with a mild vulnerability, but which plays a critical role in an important business service. A second deficiency of the CVSS metrics is that it does not account for multi-stage attacks which exploit vulnerable components to launch attacks on components deeper within the network that are not directly accessible to the attack source.

Our goal is to utilize the CVSS base scores to define and compute environmental metrics for components within an enterprise. The requirement is that the metric be intuitive and efficiently computable. The metric we define in this paper meets both requirements and captures both the difficulty of launching an attack on a component, and the impact that a successful attack can have by opening up exploits on components downstream from the component. We demonstrate scalable algorithms to compute our metrics, and give examples on enterprise-scale networks with several thousand components.

Our metrics can be used to prioritize vulnerabilities, so that system administrators need focus only on those vulnerabilities that have the most significant impact on their business. Additionally, our methods can also be used for "what-if" analyses to track changes in security levels as changes are made to applications and networks.

2 Related Work

A number of recent papers address the problem of evaluating network vulnerability. The closest work in spirit to ours is the NetSpa system [4, 5]. Similar to our approach, NetSpa also computes the reachability matrix of a network, albeit using somewhat different techniques. The main difference is our definition of the impact metric based on least-effort attack paths which captures multi-stage attacks in an intuitive manner.

Singhal and Ou [10] suggest treating exploit metrics, such as CVSS scores, as probabilities, but does not define an associated probability space, nor does it justify assumptions in treating these metrics as probabilities.

We interpret the CVSS exploitability score from Section 3.2.1 of [1](not Section 2.2.1) as a measure of effort rather than as probabilities, the latter being hard to justify for various reasons. While we remain agnostic about the validity of the CVSS scoring scheme itself, we take the operational view that since many IT organizations are comfortable with the CVSS scores, it makes sense to build on them.

Less closely related is the work on efficient generation of attack graphs [6, 7, 8, 9, 10]. These do not, however, explicitly model network router and firewall configurations to calculate the end-to-end reachability matrix, and do not focus on the problem of defining an aggregate impact metric.

A number of vendors, such as Qualys Guard, Red Seal Systems, and Skybox offer vulnerability assessment products. Some of the publicly available literature on these products claim to use "attack paths" in their assessment calculations, but beyond that it is unclear exactly how their methods work.

3 Defining Environmental Metrics

We build on the CVSS metrics; our innovation is that our metrics also account for the topological interconnections between components within the enterprise network. This builds on previous work [2].

Another innovation of our work is that we account for the possibility of multi-stage attacks. An attacker can launch an attack on any component if it has end-to-end connectivity to the component. Once a component vulnerability has been exploited and root access gained on the underlying machine, the attacker can launch further attacks inside the enterprise. Such multi-stage attacks can have devastating impact because exploited machines deep within the network generally have greater network connectivity to internal machines than available directly to the attacker.

Each application component can have multiple vulnerabilities. For each vulnerability, CVSS assigns an *exploitability* metric that captures the level of difficulty of exploiting the vulnerability. For this paper, we scale the CVSS exploitability metric to a number in the range [0,1]; smaller exploitability scores indicate a less easily exploited vulnerability while a higher value indicates a more easily exploited vulnerability.

In addition, we require that every component application be assigned a *criticality* metric by an IT administrator; this metric can, for example, be based on the relative criticality of business services that depend on this component. For example, a service, such as corporate email service, a corporate portal, or a general ledger system is made up of multiple applications. Our assumption is that an IT organization can assign a criticality metric to each of the services it provides that reflects the business priorities of the business functions supported by each service. For example, we may determine that the accounts receivable

service is more important than the employee portal. Identifying services and their associated priorities must be done manually by IT and business units. We assume that the criticality metric is a number 0 or greater; larger criticality scores indicate components that are more critical to the enterprise.

3.1 Exposure and Impact

For any component within an enterprise, there are two fundamental metrics that we wish to capture:

- Exposure: how easy is it for the adversary to exploit the component vulnerability?
- Impact: what is the aggregate criticality of all the vulnerabilities that can be exploited by the adversary?

To measure the exposure of a component, we consider all possible attack paths from the adversary to the component. In order to do this, we first compute the reachability graph of the enterprise.

Our abstract graph-theoretic model of an enterprise includes a node $< a, m >$ for every application component a deployed on a machine m. We include a directed edge from node $< a, m >$ to every node $< a', m >$ corresponding to applications that share the same underlying machine. We also include directed edges from $< a, m >$ to $< a', m' >$ if the port corresponding to application a' on machine m' is reachable from machine m within the network. This calculation is enabled by previous work, for example [2] which demonstrated how all end-to-end connectivities can be computed efficiently from the configurations of network routers and firewalls. Reachability calculations along similar lines, with some minor differences, also appear in [4, 5].

Each component a can have multiple vulnerabilities. Let $E_{i,a,m}$ denote the CVSS exploitability score for vulnerability i that is associated with component a and deployed on machine m. We define the quantity $W_{i,a,m} = 1/(E_{i,a,m} + \epsilon)$, where $\epsilon > 0$ is chosen to be extremely small (essentially to rule out the possibility of division by 0).

The quantity $W_{i,a,m}$ can be interpreted as the work required of an adversary to exploit the vulnerability, assuming that the adversary has direct access to the vulnerable application.

When the adversary does not have direct access to a component, it may still have indirect access. The attacker can launch an attack on any component if it has end-to-end connectivity to the component. Moreover, once a component vulnerability has been exploited and root access gained on the underlying machine, the attacker can launch further attacks inside the enterprise. Such multi-stage attacks can have devastating impact because exploited machines deep within the network generally have greater network connectivity to internal machines than available directly to the attacker.

We capture multi-stage attacks as paths in our graph from the adversary to the component. In this paper we restrict attention to vulnerabilities whose successful

exploit results in privilege escalation, i.e., root access. Thus, an adversary can launch an indirect, multi-stage attack by attacking components one-by-one along the path. Of course, launching multi-stage attacks requires additional work, and our goal is to capture the extra work in an intuitive way.

Consider a path $p = v_0, ...v_k$ from the adversary v_0 to a component v_k. Let W_i be the work required of the adversary to successfully exploit component v_i when he has direct access to v_i. We define W_p the work required along path p as:

$$W_p = W_1 + cW_2 + c^2W_3 + \cdots + c^{k-1}W_k,$$

where $c > 1$ is a constant amplification factor.

The amplification factor captures the additional step in a multi-stage attack amplifies the work of an adversary to attack downstream nodes. The intuition behind path weight is that the likelihood of a multi-stage attack depends not only on the exploitability of the intermediate nodes, but also on the length of the path; multi-stage exploits need to be much more sophisticated, and therefore require more work, to succeed.

Next, we define the exposure of a component v_k as $exp(v_k) = \frac{1}{min_p W_p}$, the minimum work path, over all paths from the adversary to the component. This captures the intuition that any component is as exposed as the weakest attack path to it from the adversary.

Note that the exposure metric can be efficiently computed for large networks using well-known shortest path algorithms.

Finally, the impact of an adversary on a network is defined as

$$I(A, N) = \sum_{v:nodes \in N} exp(v) * C(v),$$

where $C(v)$ is the criticality of node v. Using this measure as opposed to *Collateral Damage Potential* in Section 2.3.1 of CVSS [1] has the advantage of accounting for multistage attacks.

3.2 An Application: Prioritizing Patches

Suppose that we have computed the exposure of all nodes in a network, and the impact of an adversary. How do we use these metrics to prioritize vulnerable components for patching?

A simple method is to compute, for each patch, the reduction in adversarial impact if the patch were applied. This requires recalculating all the shortest paths and adversarial impact for each patch, and choosing the one which reduces the impact by the largest amount. This process can be iterated in a greedy manner to prioritize all patches.

Besides prioritizing patches, the metrics can also be used to evaluate alternative designs to update the network and applications.

4 Experimental Results

We have implemented our techniques above; specifically, in addition to algorithms for computing the metrics outlined, we automatically scrape CVSS feeds

to gather exploitability metrics used in our calculations. On networks with several thousand nodes, the algorithm to compute the metrics take under one minute. This supports our view that our techniques can be useful for performing what-if analyses as part of change management planning, and for re-evaluating risks in response to new CVE reports in real time.

We have also run tests, described below, to validate that the scoring functions are consistent with intuition. In an initial test on a simplified scenario, the metrics indicated that a component with lower CVSS vulnerability was given higher patching priority than another component with a higher CVSS score. This was indeed correct, as the component with lower CVSS score was easier to exploit and, unlike the other component, was upstream of critical components.

4.1 Network Description

The networks we tested our ideas on were synthetically generated from a template; the topology and applications were designed to be representative of large enterprise networks.

The network template has components consisting of one adversary, and end-users connected to a mirrored infrastructure. Each mirror consists of a 3-tier architecture (replicated copies of web, application, and data base servers), and different categories of administrator machines connected via a separate administrator network. The number of end-users, mirrors, and administrators and servers within each mirror are parameterized so we can easily scale the sizes of the networks for our purposes. For example, for the experiments reported here, each mirror site consisted of fifty data base servers, twenty five application servers (ASA, ASB), and twenty five web servers(WSA, WSB). These mirrors were administered by three categories of administrators with ten(admS), twenty(admDb) and five machines(admAll) respectively. The number of end-users was one thousand.

A schematic diagram of the network is depicted in Figure 1. The network has three compartments: (a) the end-users network, (b) the infrastructure network and (c) the administrative network. The connectivity (number of reachable pairs) between nodes varied by experiment; a typical set of numbers is shown in Table 1. In this table each cell (other than in the first row or first column) contains the number of edges that connect the source component(the first cell of its row) with the destination component (first cell of its column).

For a fixed choice of the connectivity matrix, we generated 25 test networks, with randomly chosen edges according to the connectivity requirements. Thus, each data point in our result corresponds to 25 runs.

This network schema is meant to represent the network structure of a medium size enterprise. It captures the kind of compartmentalization and connectivity encountered in in-house networks. The connectivity numbers can be adjusted as desired. The adversary is a single node with connectivity to the internal parts of the enterprises – all of them, and varied by experiment.

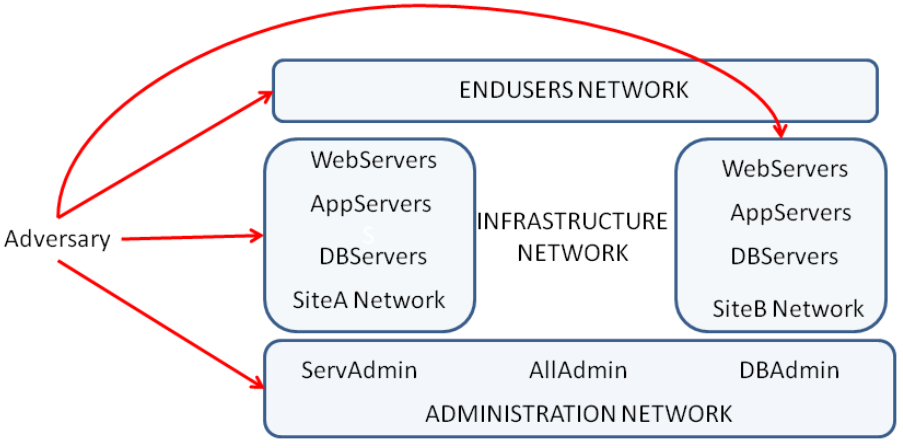


Fig. 1. Schema for synthetic experimental networks

Table 1. Typical connectivities for the synthetic test networks

	End Users	WSA	WSB	ASA	ASB	DBA	DBB	admS	admDb	admAll
Adversary	1 - 10	1-10	1-10					1		
EndUsers	100k	40k	40k	1k	1k					
WSA	500			50	50	50	50	50		
WSB	500			50	50	50	50	50		
ASA		250	50	250	50				200	25
ASB		50	250	50	250				200	25
DBA		250	50	250	50				200	25
DBB		50	250	50	250				200	25
admS	2000			500	500			50	50	5
admDb	1000					500	500	10	50	5
admAll	5000	125	125	125	125	250	250	100	50	25

4.2 Experimental Approach

Our goal is to validate our metric by showing that the formalism yields plausible results for a variety of networks that resemble enterprise networks under our assumptions about how malware propagates and how critical administration deem their resources such as server and end user machines to be.

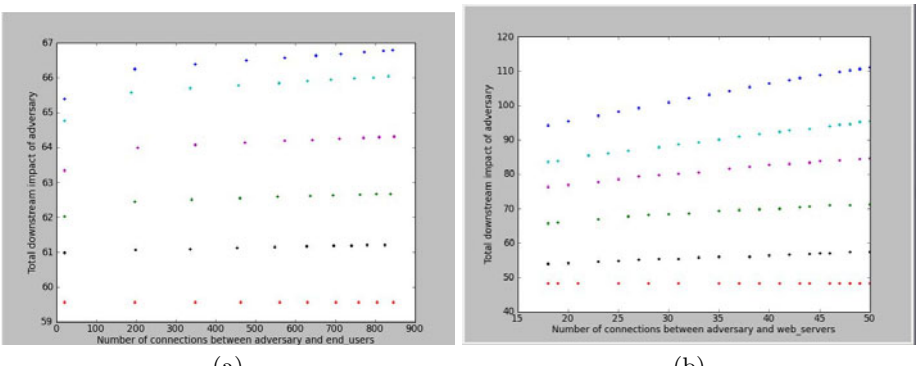
In particular we want to show that the downstream impact of the adversary varies with respect to input parameters in a manner consistent with our intuition. The specific question we address is: how many end-user machines must an adversary be able to directly exploit in order to have the same impact that he can achieve by exploiting one web server directly?

The experiment was to measure the downstream impact of varying the number of edges from the adversary to (a)end user machines versus (b) web servers. This experiment was carried out for various average values of *exploitability* of end user machines and various average values of exploitability of web server machines.

In generating our networks we specified the average criticality of different component types. For example, while the average criticality of an end user node was 0.202, individual end user node criticialities deviated from the mean randomly. For completeness, the criticalities of all the component types were set as follows – endUsers:0.20239, WSA:85.68, WSB:85.44, ASB:83.56, ASA:86.0, DBA:843.6, DBB:848.8, adminDB:677.0, adminS:724.0, adminAll: 743.0.) Thus, for example, the average web server was 425 times more critical than an end user node. Choosing different numbers affected the magnitude, but not the nature of the results.

We carried out two series of experiments – one in which the number of edges from the adversary to the end user was varied in steps of 100 and another in which the number of edges between the adversary to the web servers was added in steps of 2. In Figure 2(a) the average user exploitability takes values from the set {0.01, 0.21, 0.29, 0.42. 0.63, 0.76}. In Figure 2(b) the average web-server exploitability takes the values from the set {0.002, 0.133, 0.26, 0.47, 0.67, 1.0}. As noted earlier, for each choice of parameters and connectivity, we computed the metrics over 25 networks generated at random according to the connectivities chosen.

For running the experiments we chose $c = 1.25$. Changing the value of c between 1.1 and 1.75 did not significantly alter the nature of the results.



(a) (b)

Fig. 2. Changes in downstream impact of adversary as a result of changing the number of edges from the adversary to (a)End user machines and (b) Web servers. The implicit family of curves in (a) and (b) represent the effect of changing average exploitability – in (a) the average user machine exploitability takes values from the set {0.01, 0.21, 0.29, 0.42. 0.63, 0.76}, and in (b) the average web server exploitability takes values from the set the set {0.002, 0.133, 0.26, 0.47, 0.67, 1.0}.

4.3 Results and Their Interpretation

The graphs in Figure 2 show how increasing the exposure of web servers has much greater impact than increasing the exposure of end user nodes. Each experiment was run 25 times with the edges between the adversary and web-servers or end-user machines generated at random. Each dot in the figure is actually a composite of twenty five values, one from each run.

In the first graph the impact saturates. This can be explained by the fact that end user nodes are not particularly highly connected. The advantage gained by the adversary by connecting to more end users quickly tapers off. On the other hand web servers are well connected and hence the impact scales linearly.

The graphs can also answer the question: How much should the average vulnerability of a web-server be decreased in order to bound adversarial impact in case of a web server compromise to a desired level? From these figures it can be seen that to have the same downstream impact as increasing the connectivity between the web server and the adversary by about ten connections, it takes roughly two orders of magnitude of increased connectivity between the adversary and the end users.

This is in line with our intuitive expectations. Firstly, the web servers are more critical. Secondly, they are well connected to other critical components such as application servers and database servers. Thus our metric quantifies the advantage the adversary gains by connecting to web servers as opposed to end users.

5 Conclusions

In this paper we proposed a new way to define the CVSS environmental metric. The definition we propose can be efficiently computed, and the experiments show that the metric behaves in an intuitive way as network parameters are varied. Further evaluation on large-scale production networks is needed to see if these ideas can be fruitful in practice.

In the future, we anticipate that such metrics could be computed by integrating the analysis algorithms (reachability and metric evaluation) with an asset management system such as a Universal CMDB (Common Management Data Base), to get access to each of the components and their configurations,and with CVE feeds, for example from the National Vulnerability Database.

References

1. Common Vulnerability Scoring System, http://www.first.org/cvss/cvss-guide.html
2. Bandhakavi, S., Bhatt, S., Okita, C., Rao, P.: End-to-end network access analysis. In: HP Laboratories Technical Report HPL-2008-28R1. HP Labs (2008)
3. Forbath, T., Kalaher, P., O'Grady, T.: The total cost of security patch management. Technical report, Wipro Technologies (2005), http://wipro.com/webpages/insights/security_patch_mgmt.htm

4. Ingols, K., Chu, M., Lippmann, R., Webster, S., Boyer, S.: Modeling modern network attacks and countermeasures using attack graphs. In: ACSAC 2009: Proceedings of the 22nd Annual Computer Security Applications Conference. IEEE Computer Society, Washington, DC, USA (2009)
5. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: ACSAC 2006: Proceedings of the 22nd Annual Computer Security Applications Conference. IEEE Computer Society, Washington, DC, USA, pp.121–130 (2006)
6. Noel, S., Jajodia, S.: Optimal ids sensor placement and alert prioritization using attack graphs. Journal of Network and Systems Management 16 (2008)
7. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 336–345. ACM, New York (2006)
8. Pamula, J., Jajodia, S., Ammann, P., Swarup, V.: A weakest-adversary security metric for network configuration security analysis. In: 2nd ACM Workshop on Quality of Protection. ACM Press, New York (2006)
9. Sawilla, R.E., Ou, X.: Identifying critical attack assets in dependency attack graphs. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 18–34. Springer, Heidelberg (2008)
10. Singhal, A., Ou, X.: Techniques for enterprise network security metrics. In: Cyber Security and Information Intelligence Research Workshop. ACM, New York (2009)

A Kolmogorov Complexity Approach for Measuring Attack Path Complexity

Nwokedi Idika and Bharat Bhargava

Purdue University, Department of Computer Science
305 North University Street,
West Lafayette, IN 47907 USA
{nidika,bb}@cs.purdue.edu

Abstract. The difficulty associated with breaching an enterprise network is commensurate with the security of that network. A security breach, or a security policy violation, occurs as a result of an attacker successfully executing some attack path. The difficulty associated with this attack path, then, is critical to understanding how secure a given network is. Currently, however, there are no consistent methods for measuring attack path complexity that make the assumptions of a modeler explicit while providing flexibility in how the modeler models the attack path. To provide these desirable attributes, we propose a regular-expressions-inspired language whose rationale for attack path complexity measurement is based on Kolmogorov Complexity. After detailing our Kolmogorov Complexity-based method, we demonstrate how it can be applied to a novel security metric: the K-step Capability Accumulation metric–a metric that defines the security of a network in terms of the network assets attainable for attack effort exerted.

Keywords: security, security metrics, attack graphs, exploitability.

1 Introduction

A completely secure network is one where no attacker can violate a security policy of that network. Since such a system is currently impractical, an approximation to it would be one where the attacker has extreme difficulty violating the network's security policies. Thus, one way of asking "how secure is this network?" would be to ask "how *difficult* is it to violate a security policy in this network?" When violations occur, they happen as result of a weakness, or a series of weaknesses, that the attacker leverages. These weaknesses in the network are referred to as vulnerabilities. The complexity associated with exploiting a vulnerability is then critical to the security of a network.

The ease with which an attacker can successfully take advantage of a vulnerability is referred to as exploitability. The importance of exploitability is reflected by its inclusion in all the well-known vulnerability scoring systems. For instance, the Common Vulnerabilities Scoring System (CVSS) [1] computes exploitability as a linear combination of qualitative values. The Computer Emergency Response Team/Coordination Center (CERT/CC) [2] produces a numeric

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 281–292, 2011.

vulnerability score based on a series of questions–one of which is, "How easy is it [the vulnerability] to exploit?" The SANS Critical Vulnerability Analysis Scale Rating [8] includes the ease of exploitation in defining two of its four ratings.

Security metrics that rely on exploitability measure security in terms of the effort or skill the attacker needs to cause a security policy violation. When security violations occur as a series of vulnerability exploits, the dependencies among these exploits can be incorporated into security metrics that utilize the attack graph. An attack graph is an abstraction that reveals the ways an attacker can use identified vulnerabilities interdependently in a network to violate a security policy. In this paper, we use condition-oriented attack graphs [9] where nodes correspond to hosts, and edges correspond to vulnerabilities. Examples of attack graphs are displayed in Figure 3. Security metrics based on attack graphs are referred to as attack graph-based security metrics. An instance of such a metric would the Shortest Path metric. This metric [4] denotes the security of a network as the path from the attack graph that is of least of resistance. Another example attack graph-based security metrics is the Average Path Length metric. This metric [10] denotes the security of a network as the mean difficulty associated with all paths from the attack graph.

In the attack graph, attack path difficulty (or complexity) is represented as path length. Path length may be represented as the number of edges between the attacker's initial state and goal state or as a more complicated algebraic permutation of vulnerability scores along attack paths. While such flexibility in path length representation may be appropriate and necessary for a network administrator to model the threat of a given network, such flexibility hinders consistent communication across organizations and among researchers. If a method more complex than simply counting the edges along a path is used, some qualitative information is being employed. Because quality can be subjective, different organizations and researchers may score vulnerabilities differently and potentially compute path length differently. These differences are fine as long as the assumptions being made are explicit and grounded in theory.

We propose an approach to maintain flexibility in how attack path complexity is modeled, make associated modeling assumptions explicit, and provide a theoretical basis for attack path measurement. Our approach is called the Kolmogorov Complexity Method (KCM). KCM allows attack paths to be modeled quantitatively or qualitatively using a regular-expressions-inspired language. After describing this methodology, we show how these methods can be applied to a novel security metric called the K-step Capability Accumulation (KCA) metric.

The remainder of the paper will have the following organization. Section 2 describes KCM. Section 3 specifies the KCA metric and how KCM can be used with the metric. Section 4 gives related work associated with attack path complexity and with our novel attack graph-based security metric. Section 5 details our conclusion.

2 Kolmogorov Complexity Method (KCM)

Given an alphabet and the infinite number of strings that can be produced from that alphabet, some strings are more complex than others. Kolmogorov

Complexity is an approach for determining string complexity. Thus, given two strings, Kolmogorov Complexity determines which string is more complex.

Kolmogorov Complexity determines a string's complexity using the size of the smallest program that can produce that string [5]. Let K represent the Kolmogorov Complexity function. In comparing two strings, x_1 and x_2, if $K(x_1) < K(x_2)$, then x_2 is more complex than x_1, because a larger program is needed to describe x_2. The use of Kolmogorov Complexity to monitor security was first promulgated by Evans et al. in [6]. Kolmogorov Complexity has also been applied to spam filtering [14] by Spracklin and Saxton. We are applying Kolmogorov Complexity to security assessment–and more specifically, the complexity of attack paths.

Kolmogorov Complexity provides a systematic approach for determining complexity of attack paths. There is, however, a caveat to its use. Ming et al., in [5], have shown that determining a lower bound K is impossible. With respect to attack path complexity, this finding means that we can never find the "true" effort needed for any attacker to exploit a vulnerability. This constraint need not be a hindrance to its use if reasonable effort estimates can be obtained. Nonetheless, KCM provides a standard way for communicating attack path complexity that is based upon a sound theory of complexity. Such formalism provides consistency among researchers when comparing measurements. Below, we provide an attack path complexity description language inspired by the language of regular expressions.

Alphabet

1. A corresponds to the exploits (i.e., instances of vulnerabilities) found in all attack graphs being considered.

Constants

1. ϵ corresponds to the empty string.
2. $e_i \in A$ denotes an exploit from one of the attack graphs being considered.
3. $\emptyset$ corresponds to the empty set.

Operations

1. Let S and T be two strings comprised of characters from A.
2. Let E_1 and E_2 be expressions of the language.
3. ST evaluates to the concatenation of strings S and T.
4. () provides priority ordering of evaluation.
5. $(S)^+$ states that the expression S *may* repeat one or more times but must appear once.
6. S^k evaluates to k instances of S concatenated together.
7. $E_1^{[k]}E_2$ evaluates to inserting E_1 at index k in E_2, where the first character in E_2 corresponds to the zeroth index. This can be generalized to $E_1^{[k_1],[k_2],[k_3],\ldots,[k_{n-1}],[k_n]}E_2$. Note: In general, this rule would not be used independently. It would only be used as part of the following two rules.
8. $E_1^{l,[k]}E_2$ evaluates to concatenating E_1^l to E_2, and inserting E_1 into index k of E_2.
9. $E_1^{l[k]}E_2$ evaluates to inserting E_1^l into index k of E_2.

The above KCM language is flexible in its ability to model the complexity of attack paths. For instance, the attack path in Figure 1 can be represented multiple ways. H1 through H6, Attacker, and Target in Figure 1 correspond to hosts, and v_j corresponds to vulnerability j. Based on operations 8 and 9 from the KCM language, a qualitative representation, KCM-qual, of this path can be represented as $v_1^{3,2[2]}v_2v_3$ yielding an attack path length of 3 vulnerabilities. An equivalent representation would be $v_1^{3,[2],[2]}v_2v_3$ yielding an attack path length of 3 vulnerabilities. In both representations, E_1 and E_2 from operations 8 and 9 corresponds to v_1 and v_2v_3 respectively. This representation suggests that once the attacker exploits H1 via v_1, attacking hosts H2, H3, H6, and Target is trivial because they, for example, require the same credentials to have their vulnerabilities exploited. Such a representation may also be appropriate if known scripts can be used to exploit vulnerabilities in the system. Ultimately the qualitative representation used depends on the subjective decision of the modeler. For instance, the attack path can alternatively be represented as $v_1^{3,[2]}v_2v_3v_1$. The path length of this representation is 4 vulnerabilities. This representation suggests that the attack path in Figure 1 is more secure than what is suggested by the two representations initially described for this attack path. The semantics of this latter representation suggests that while the same information can be used in exploiting hosts H2, H3, H4, and H6, different information is required to exploit Target. If the attack path is represented as $v_1v_1v_1v_2v_3v_1v_1$, then this would, for instance, suggest that different credentials are required for compromising each host. This representation corresponds to the quantitative representation, KCM-quant, which is equivalent to counting the edges along attack paths.

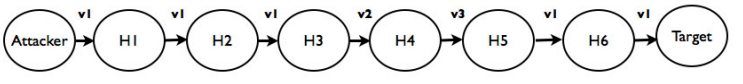


Fig. 1. A single attack path with Attacker, H1–H6, and Target corresponding to hosts and v_j corresponding to vulnerabilities

2.1 Representing Cycles in Attack Graphs

It is possible for an attack graph to contain cycles. A cycle in an attack graph corresponds to an infinite number of paths. Thus, it can be difficult to reason about such attack paths (e.g., determining attack path length). Typically, cycles are ignored when performing attack graph analysis. Using KCM-qual, a modeler can account for the path length of attack paths containing cycles.

Figure 2 shows a infinite number of attack paths because there is a cycle. Using KCM-qual, we can represent such attack paths. The infinite number of attack paths appearing in Figure 2 can be, for example, captured with $v_1^2(v_1v_2v_3)^+v_1^2$. The attack path length of this cyclic attack path is 5 vulnerabilities.

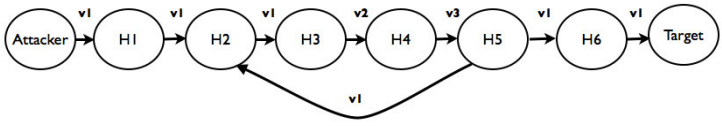


Fig. 2. An attack path containing a cycle

2.2 Qualitative versus Quantitative Representations

The decision of whether to use the qualitative representation of KCM, KCM-qual, or the quantitative representation, KCM-quant, is predicated on the amount of information available for the security assessment. If the security engineer has no information regarding exploitation difficulty, KCM-quant is appropriate. KCM-quant assumes all vulnerabilities require distinct, yet, equal amounts of effort. This assumption is overly restrictive in practice. Therefore, practitioners may find KCM-qual most amenable for measuring attack path complexity.

If the security engineer has exploitability information about the vulnerabilities in the network under inspection, then KCM-qual would be appropriate. If the security engineer knows that two vulnerabilities require the same exact actions and information to be exploited, these exploits can be coalesced into a single vulnerability in KCM-qual. An example of this scenario would be when the attacker uses the same credentials to access different hosts within a network. Another example of when two exploits may be coalesced in KCM-qual is when some information (e.g., host name) that the attacker has already acquired or that is publicly known is all that is required to be changed when needing to exploit the second vulnerability. If the security engineer assumes that the attacker is rational, then KCM-qual corresponds more closely to what the attacker will actually experience.

The assumptions the modeler uses can be reflected in the names used for different qualitative representations. KCM-qual-C corresponds to the KCM-qual representation where nodes in an attack path are coalesced *only* when the same credentials can be used at each node to realize an exploitation. KCM-qual-S corresponds to the KCM-qual representation where nodes in an attack path are coalesced *only* when there exists some publicly accessible software that reduces the effort of vulnerability exploitation at each node. Under KCM-qual-S, even if two nodes require different software, each node would be coalesced. However, under KCM-qual-S$^-$, a less conservative formulation of KCM-qual-S, nodes would only be coalesced if the same software could be used to exploit vulnerabilities at different nodes. KCM-qual-CS corresponds the combination of KCM-qual-C and KCM-qual-S. Under KCM-qual-CS, nodes on an attack path will be coalesced if with KCM-qual-C holds or KCM-qual-S holds. When KCM-qual is used in this paper, it is referring to the KCM-qual-CS formulation. Naturally, there is also a KCM-qual-CS$^-$. KCM-qual-CS$^-$ will coalesce two nodes on the same attack path if either KCM-qual-C holds or KCM-qual-S$^-$ holds.

If truly quantitative complexity or probability values were known for each distinct vulnerability in the attack path, then these values could be incorporated

with KCM. By assigning these values to the nodes in the KCM-quant representation, algebraic operators would be available to manipulate the values along these paths. If the KCM-qual representation is used, algebraic operators would be unavailable for manipulating the values along attack paths as the result would be mathematically unsound. This soundness issue could be practically bypassed if empirical results showed KCM-qual to be in closer alignment with reality than KCM-quant. The problems of determining the complexity of exploiting a vulnerability or determining the probability of a vulnerability being exploited are critical issues, open problems, and outside the scope of this paper.

3 K-Step Capability Accumulation (KCA) Metric

The K-step Capability Accumulation (KCA) metric specifies the "power" the attacker can obtain on a network in K steps. Practically, "power" refers to access level. Users with the highest access level have the most capability in a network. Such users also have the most opportunity for violating the network's security policy due to their level of access. The granularity of access levels ranges from system-level access to application-level access. This granulation is necessary in some instances as the attacker may successfully obtain system level access to machines on a network and yet still require access to specific authenticated applications. We show how to integrate such application-level accesses in the following subsection. These semantic variations in access level is best captured by the term "capability." This term differs from the notion of capability defined by Phillips and Swiler in [4]. In [4], capability is used broadly to include not only privileged actions within a network, but it also includes the skill level of the attacker and the tools or resources available to the attacker. The capabilities KCA is concerned with are those derived from the network under inspection, that is, privileges and privileged actions. The examples in this paper usage of access levels follows the concept of privileges in [12,13] by using system-level access on machines to represent power.

3.1 Evaluating with KCA

In determining which of two networks Sys_1 and Sys_2 are most secure using KCA, one procedure would resemble the following. First, generate attack graphs G_1 and G_2 for Sys_1 and Sys_2 respectively. Choose an integer for K that corresponds to the maximum number of steps the analysis should be carried out for if needed. K should be no greater than the minimum of the maximum path length for both attack graphs. One evaluation method is that if an attacker can obtain more capabilities on G_1 in K steps, than the attacker can on G_2 in K steps, then Sys_2 is more secure than Sys_1. KCA is given below.

$$Cap_h(G) = \cup_h capabilities(n) \tag{1}$$

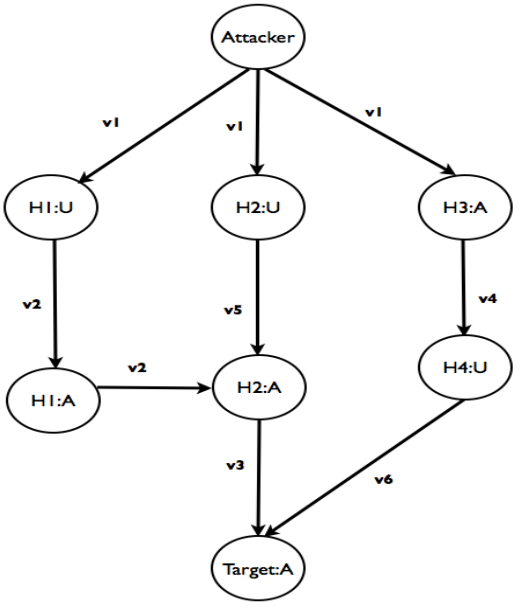
$$KCA_k(G) = \cup_{i=1}^{k} Cap_i(G) \tag{2}$$

capabilities(n) returns the set of capabilities available at node n. If more granularity is desired, the *capabilities* function may be extended with a set of vulnerabilities as an input parameter. Different vulnerabilities may provide different capabilities for an attacker. This extension may be captured by extending *capabilities*(n) as *capabilities*$(\{v_1, v_2, v_3..., v_{m-1}, v_m\}, n)$. Vulnerabilities v_1 through v_m correspond to the vulnerabilities that an attacker may exploit to compromise node n. This extended representation allows for application-level access control to be modeled. Such a model also allows for different attacker profiles. That is, not all attackers have the same skill level or resources [11]. Therefore, depending on the assumptions made about the attacker, potentially only a subset of the vulnerabilities may be exploitable. Using our extended representation this scenario can be captured. $Cap_h(G)$ represents the capabilities obtained at level h in attack graph G. h represents the distance from the attacker's initial state. $KCA_k(G)$ is then simply the union of capabilities obtained at each level up to k.

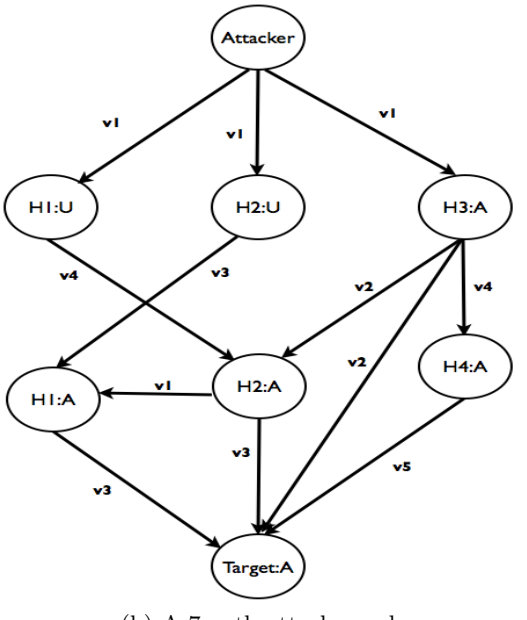
Applying KCA to the attack graph in Figure 3(a), we obtain different values at each level of the attack graph. The nodes of the attack graph correspond to hosts and access levels. The labels have the format of *host:access level*. U corresponds to user-level access. A refers to administrator-level access. If access level is not specified, the attacker is assumed to have administrator-level access on the machine. In computing KCA for the attack graph in Figure 3(a), we obtain the following. $Cap_1 =$ user-level access on H1 and H2, administrator-level access on H3. $Cap_2 = Cap_1$ and administrator-level access on H1 and H2, and user-level access on H4. $Cap_3 = Cap_2$ and administrator-level access on the Target host. The attacker can accumulate all privileges in the network in 3 levels.

When comparing two network configurations to decide which is most secure with respect to this metric, we have two options. One option is to compare the two networks' corresponding attack graphs only after k steps. A second option is to compare the two network's corresponding attack graphs at each level up to k. The second option is useful when a security engineer is interested in how quickly the attacker attains capabilities. For instance, after k steps in the two attack graphs being compared, the attacker may have attained the same level of capabilities. This information does not help the security engineer decide which of the two attack graphs is more secure. However, if the security engineer were to examine the two attack graphs using KCA at each level up to k the security engineer may find that in one attack graph, the attacker accumulates all of the network's capabilities after 1 step, whereas in the other attack graph the attacker accumulates all of the network's capabilities after 5 steps. This additional information would suggest to the security engineer that the network corresponding to the latter attack graph is more secure because it requires the attacker to take more steps to acquire all of the network's capabilities.

An alternative method for comparing two network configurations using KCA would be to compare KCA at each step. That is, in comparing two networks, the security engineer deems the network whose attack graph allows for the same or more valuable assets attainable in fewer steps to be less secure. This method of comparison is best exemplified through an example. Observe the attack graph



(a) A 3 path attack graph



(b) A 7 path attack graph

Fig. 3. Two attack graphs

in Figure 3(b). The KCA at level 1 for the attack graph in Figures 3(a) and 3(b) is the same. However, we can see that in Figure 3(b) that Target can be reached in two steps. Assuming that this is the most important host to protect, then the attack graph in Figure 3(b) is less secure because the attacker is able to obtain more power in fewer steps.

3.2 Applying Kolmogorov Complexity Method to KCA

The KCA metric uses steps or levels within the attack graph to obtain its values. The KCA metric as previously defined utilizes KCA-quant. To accommodate KCA-qual, we need to redefine *Cap*.

$$Cap_{p_i} = \cup capabilities(p_i) \tag{3}$$

This path-oriented version of Cap extracts all capabilities along an entire path p_i.

A step or level in KCA under KCM-qual uses the characters in the string representation to determine levels. If an attacker exploits a vulnerability v_j that is followed immediately by $0 \leq m < k$ v_j's, then the attacker has the capabilities associated with each host that v_j violates. If any new vulnerability v_l is encountered on the path, then for any v_j or v_l immediately following this vulnerability the attacker has the capabilities associated with each host that is violated by v_l or v_j. This pattern continues when evaluating KCA with KCM-qual. If using a KCM-qual representation that involves KCM-qual-S, then any vulnerabilities the software enables exploitation for can belong to a vulnerability superclass that would allow these vulnerabilities to be coalesced.

Let s_1 to s_n correspond to the KCM-qual string representations of attack paths p_1 to p_n. Let q_1 to q_n correspond to the KCM-quant string representations of attack paths p_1 to p_n. Let $q_j^{0..i}$ correspond to the substring of q_j that includes indices 0 through i. A similar definition exists for s_j. Let s_j^i correspond to the ith character position in s_j. A similar definition exists for q_j. Let $e(s_j^{0..i}) = q_j^{0..m}$, for some integer m such that the vulnerability at index s_j^i and q_j^m are the same, and the vulnerability does not appear in q_j^{m+1}, and the vulnerabilities present before q_j^m all appear in $s_j^{0..i}$. Then the equation for KCA using KCM-qual is given by the following:

$$KCA_k(G) = \cup_{i=1}^{k} Cap_{e(s_j^{0..i})}(G), \tag{4}$$

for all attack paths j.

This manifestation of KCA captures the fact that an attacker may obtain knowledge from one attack path and apply this knowledge to a completely separate attack path. If this version of KCA is applied to Figure 3(a), we must first determine how to represent each path. We will then define each path from left to right as the following: $v_1 v_2^2 v_3$, $v_1 v_5 v_3$, $v_1 v_4 v_6$. Thus, after one iteration, the attacker will have H1:U, H2:U, and H3:A. After two iterations, H1:A, H2:A, and H4:U will be unioned with the attacker's previous capabilities. Note that the attacker is able to obtain H2:A through both v_2 and v_5. Both are accounted for in this iteration. In the final iteration Target:A is added to the capabilities of the attacker.

4 Related Work

Because this work proposes an attack path complexity measurement method *and* a security metric, this section has two parts. The first subsection is dedicated to attack path complexity. The second subsection is focused on relevant attack graph-based security metrics.

4.1 Attack Path Complexity

In the authors' literature search, attack path complexity is a topic that has not been directly addressed. An attack path's length can simply be the number of edges (i.e., vulnerabilities) between the attacker's initial state and goal state [4]. If complexity values are assigned to each vulnerability in the form of probabilities, then the product of a given attack path would correspond to the complexity of that attack path [4]. Complexity may also be summed [3,4].

Wang et al., in [3], noted that a relation could exist between two vulnerabilities such that the exploitation of one vulnerability decreases the complexity in exploiting the other vulnerability. KCM-qual captures this notion as the modeler is given the ability to show that exploiting one vulnerability decreases the difficulty of exploiting other vulnerabilities to zero with use of our language (e.g., operation 6 from the KCM language). The relation in [3], allows for more flexibility within the proposed framework in that the complexity of other affected vulnerabilities can change to arbitrary values. We believe that this flexibility, unfortunately, lends itself to the type of subjectivity that hinders the sharing of security metric data. With no rules for how vulnerability complexity values should change due to a vulnerability exploitation, there can be no expectation that researchers will assign changes in complexity values in any uniform way.

4.2 Attack Graph-Based Security Metrics

There are two security metrics that have inspired KCA: the Shortest Path metric and the Network Compromise Percentage (NCP) metric. If the Shortest Path metric [4], from Phillips and Swiler, is being used under KCM-quant, then the shortest attack path in the attack graph corresponds to the path with the fewest number of edges. If KCM-qual is used, then the shortest path in the attack graph corresponds to the path that produced, through arithmetic/algebraic manipulations, the value considered to have the least resistance in comparison to other paths.

KCA and the Shortest Path metric can be similar when using a goal-oriented attack graph. However, KCA can be applied to attack graphs with no goal states. The Shortest Path metric, on the other hand, cannot be applied to attack graphs with no goal states. Thus, KCA is more versatile in its applicability to different types of attack graphs.

When there is a goal state and the semantics of the attack graph are such that this goal state has all of the asset value in the network, the KCA metric may degenerate to the Shortest Path metric. For instance, if the attacker can reach the goal state in single step and the non-attacker nodes are of little value with

respect to the target node, then using the Shortest Path metric without KCA would be sufficient for determining which of the two networks is most secure. However, if other nodes are perceived as being relevant to the network's security, then the KCA metric can be used to obtain more information about the security of the network than what the Shortest Path metric can provide.

The NCP metric [7] proposed by Lippmann et al. denotes the security of the network as the percentage of compromised hosts within the network. A NCP of 100% means that the entire network can be compromised by the attacker. A NCP of 0% means that none of the network assets can be compromised by the attacker. NCP can be extended/modified to include assets that are more fine-granular than hosts, and therefore has the same representational ability as KCA. Thus, KCA's differentiating feature is its inclusion of attack effort exerted.

5 Conclusion

Measuring network security in terms of the difficulty experienced by an attacker in attempting to violate a security policy is an intuitive perspective. However, without systematic and flexible methods for modeling the complexity associated with any given attack path, such a perspective will have limited value. In this paper, we have proposed the well-founded theory of Kolmogorov Complexity to serve as a foundation for measuring attack path complexity. We have also proposed a novel security metric that specifies network security in terms of the assets gained for the attack effort expended.

References

1. Mell, P., Scarfone, K., Romanosky, S.: Common Vulnerability Scoring System. IEEE Security and Privacy 4, 85–89 (2006)
2. Computer Emergency Response Team (CERT), http://www.cert.org
3. Wang, L., Singhal, A., Jajodia, S.: Measuring Overall Security of Network Configurations Using Attack Graphs. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 98–112. Springer, Heidelberg (2007)
4. Phillips, C.A., Swiler, L.P.: A Graph-based System for Network-vulnerability Analysis. In: Proceedings of the 1998 Workshop on New Security Paradigms, pp. 71–79. ACM, New York (1998)
5. Ming, L., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer, Heidelberg (1997)
6. Evans, S., Bush, S., Hershy, J.: Information Assurance Through Kolmogorov Complexity. In: DARPA Information Survivability Conference and Exposition (2001)
7. Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., Cunningham, R.: Validating and Restoring Defense in Depth Using Attack Graphs. In: Military Communications Conference (2006)
8. SANS, http://www.sans.org/newsletters/risk/

9. Noel, S., Jajodia, S.: Managing Attack Graph Complexity Through Visual Hierarchical Aggregation. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 109–118. ACM, New York (2004)
10. Li, W., Vaughn, R.: Cluster Security Research Involving the Modeling of Network Exploitations Using Exploitation Graphs. In: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and Grid Workshops (2006)
11. Dantu, R., Kolan, P.: Risk Management Using Behavior Based Bayesian Networks. In: Kantor, P., Muresan, G., Roberts, F., Zeng, D.D., Wang, F.-Y., Chen, H., Merkle, R.C. (eds.) ISI 2005. LNCS, vol. 3495, pp. 115–126. Springer, Heidelberg (2005)
12. Dacier, M., Deswarte, Y., Kaâniche, M.: Models and Toos for quantitative assessment of operational security. In: Proceedings of the 12th International Information Security Conference, pp.177–186 (1996)
13. Ortalo, R., Deswarte, M., Kaâniche, M.: Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security. IEEE Transactions on Software Engieering 25, 633–650 (1999)
14. Spracklin, L.M., Saxton, L.V.: Filtering spam using kolmogorov complexity estimates. In: Advanced Information Networking and Applications Workshops, pp. 321–328 (2007)

Extending LSCs for Behavioral Signature Modeling

Sven Patzina[1], Lars Patzina[2], and Andy Schürr[1]

[1] Real-time Systems Lab, TU Darmstadt, Darmstadt, Germany
{sven.patzina,andy.schuerr}@es.tu-darmstadt.de
[2] Center for Advanced Security Research Darmstadt (CASED), Germany
lars.patzina@cased.de

Abstract. Driven by technical innovation, embedded systems are becoming increasingly interconnected and have to be secured against failures and threats from the outside world. For this purpose, we have defined an integrated model-based development process for security monitors which requires an expressive, formally well-defined, and easy to learn behavioral signature language. In this paper, we demonstrate that Live Sequence Charts (LSCs) are adequate for the specification of behavioral signatures. To satisfy all requirements and enable compact modeling, we extend LSCs by concepts that fit well to the spirit of LSCs.

1 Introduction

Driven by technical innovation, embedded systems are becoming increasingly interconnected. Thus, they cannot be considered as being separated from the outside world, even though many of them were developed as such. Often, little attention has been paid to security mechanisms, such as encryption and safe component design for defense against attacks. Groll and Ruland [3] show such weaknesses for passive and active attacks in modern networks in the automotive domain and postulate that additional security measures are needed. Furthermore, Koscher et al. [7] identify the CAN bus protocol as a major security drawback in modern automobiles. For subsequent protection of these systems Papadimitratos et al. [12] propose secured communication in the car and the development of a secure architecture to improve privacy and security.

Even when all these proposed techniques are applied during the development of an embedded system, in the majority of cases it is impossible to eliminate all security vulnerabilities and to foresee all possible attacks. Considering huge heterogeneous systems or components, it is often economically or technically infeasible to secure them against external adversaries retroactively. Therefore, systems cannot be considered as safe, either due to unknown vulnerabilities, or due to the required integration of legacy components.

To secure such systems, Kumar [8] proposes monitoring the system at runtime, which permits the detection of attacks that exploit previously unknown errors and security vulnerabilities. The two most common approaches for this

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 293–304, 2011.

purpose are, firstly, signature-based detection, which uses predefined attack descriptions and, secondly, anomaly detection, which recognizes the faulty behavior of a system by detecting deviations from the intended behavior. However, signature-based detection is only able to detect attacks that are similar to known vulnerabilities and attack classes, which leads to a low false-positive rate, but also to an insufficient number of matches. In contrast, anomaly detection is able to reveal unknown attacks by observing their impact on the system, but suffers from a high false-positive rate. These false matches have to be handled by user interaction to evaluate the threat or by self-healing techniques to transfer the system to a secure and stable state.

Our goal is a comprehensible, model-based development process, based on the Model Driven Architecture concept, to automatically generate security monitors from a specification consisting of Live Sequence Charts (LSCs) structured by use and misuse cases. The process – depicted in Fig. 1 – starts in the requirements phase, where the intended system behavior is modeled as use cases and known attack patterns and attack classes are modeled as misuse cases [15]. These abstract

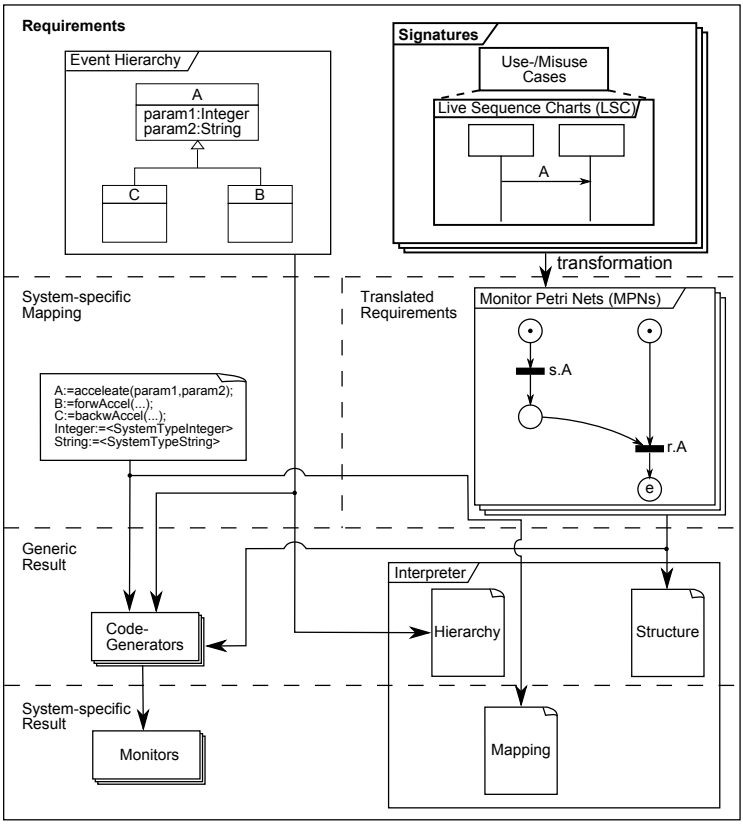


Fig. 1. Model-based security engineering process

specifications are described in more detail with LSCs. Through automatic transformation and weaving of system-specific information, a security monitor is automatically generated, using the intermediate Monitor Petri net language [13].

Our contributions in this paper are:

- the evaluation of LSCs as a signature language.
- the proposal of extensions to classic LSCs, to overcome their limitations.
- the application of extended LSCs to an example scenario.

In this paper, we show that LSCs are applicable to the description of behavioral signatures. In Sect. 2, various approaches for modeling policies, together with their advantages and disadvantages, are presented. Subsequently, the requirements that have to be satisfied by a behavior specification language are introduced in Sect. 3. According to these, we present in Sect. 4 the required conditions and extensions to satisfy the requirements for a policy language. Section 5 draws a conclusion and describes starting points for future work.

2 Related Work

The development process we present for security monitors requires a language to specify intended and forbidden behavior as signatures. This language must be able to describe functional and non-functional requirements (NFRs) that need to be monitored. Specifying NFRs is more challenging than specifying functional requirements. These NFRs are often described in a natural language and are therefore very abstract. To obtain a more formal description of these policies and behavioral signatures, special policy languages or temporal logic are commonly used for Intrusion Detection Systems (IDSs).

One of these concepts are expert systems (ES) such as CLIPS [2] and P-BEST [9]. These systems are based on inference rules with an if-then structure. If the guard of a rule is satisfied, then the specified action is performed on facts that describe the monitored system. Thus, the basis of facts is modified by the system and by the rules, which can trigger new actions. Furthermore, temporal rules have to be formulated over the facts, which is not at all intuitive. Another, better fitting, approach to formulate temporal aspects is the usage of temporal logic as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL).

These languages are often hard to learn and are not comprehensible for non-practitioners. To overcome these shortcomings, several extensions to the de facto standard of the OMG – the UML – have been proposed. A lightweight extension of the UML, called UMLsec [6], extends the system specification by security properties, such as encryption of communication. Therefore, it provides a UML profile, defining stereotypes and tagged values, to annotate the system model. In contrast, SecureUML [10] is a heavyweight extension of the UML that changes the meta model. This dialect of the UML is designed for modeling role-based access control restrictions. Both were developed for extending UML to describe non-functional security constraints in a model-based development process. However, they lack the ability to describe behavioral signatures needed for an IDS.

To close this gap, Hussein et al. [5] propose a UML profile providing stereotypes to annotate several UML diagrams. They employ use cases for the scenario description, classes for the structure of the system, and state machines to model potential steps of the behavior of the system.

Beside UML statecharts, other state/transition-oriented languages are used to model behavioral signatures. A variant with a simple structure is used by STAT [19], where states are specified by invariants, and actions and transitions are annotated by conditions, events, and actions. By omitting fork and join, a transition can only have one predecessor and one successor, to ease the interpretation. This results in an explosion of states when modeling concurrent behavior, because every permutation has to be modeled as a separate state. In contrast to STAT, IDIOT [8] uses expressive Coloured Petri nets, annotated with the general purpose functional programming language ML. This complex syntax, consisting of Petri nets and a functional programming language, leads to powerful, but hard to understand, specifications.

To ease the modeling and allow non-practitioners to understand the specification, a more high-level approach for signature modeling is desirable. So Massacci and Naliuka [11] use UML sequence diagrams (SDs) extended with linear temporal logic for modeling behavioral signatures. However, the semantics of SDs is not suited to model deontic constraints (obligation, permission and prohibition). Therefore, they use them in a not UML fashioned way, which conflicts with the use of the standard and hampers the developer in using existing UML knowledge. With the same drawbacks, Solhaug et al.[18] pursue a very similar approach, by using SDs to define policy specifications. However, they use the STAIRS semantics to interpret their SDs and state that Live Sequence Charts "could serve as an alternative for interpreting policy rules".

If LSCs[4] can be used to describe the semantics of policies, why not use them as a specification in their original semantics? In the following, we will examine what the requirements for our monitor development process are and describe how these requirements can be satisfied by LSCs.

3 Requirements

To be able to define and evaluate a signature language, it is necessary to identify its requirements that have to be met. As discussed in Sect. 2, there are different possibilities to model attacks and intended behavior. Since we use the specification language in a model-based development process that should be also understandable to non-practitioners, it has to possess a graphical syntax. Though, Live Sequence Charts satisfy this high-level requirement, we take a closer look on important requirements for a software engineering process and especially for a policy (signature) language.

In this context Smith et al. [17] propose several criteria for security and software engineering processes. Adopted to our monitor development process these are:

1 - Easy to learn. The access to the modeling language should be easy. So a flat learning curve for software developers is desirable.

2 - Comprehensible. A fundamental comprehension of the modeled systems should be possible for non-practitioners.
3 - Predictive. The modeled specifications should have a defined semantics to be able to analyze or simulate them in order to reveal non-obvious properties.
4 - Effective transition to implementation. The transitions between models up to the code generation have to be properly defined and implemented.
5 - Cost-effective. Building the model has to be less expensive than other appropriate ways of building the system.
6 - Expressive. The language has to be expressive enough to represent all kinds of key concepts without losing important details.

(1) Most of the diagram types used for our specification are known by software engineers because they are based on diagrams borrowed from the UML 2.0. So UML use cases are extended by the concept of misuse cases, which allow the modeling of unintended behavior and attacks. Class diagrams are used to build a hierarchy of messages and describe the structure of the system. Live Sequence Charts are an extension of Message Sequence Charts and UML 2 Sequence diagrams and preserve the graphical appeal and intuitiveness of MSCs [20]. This enables the developer to start modeling right away by extending his knowledge stepwise for the special properties of the languages.

(2) In contrast to many existing special policy languages like expert systems or state/transition-based approaches, where the human readability is no core issue [16], the LSCs can be easier understood by non-practitioners.

(3) LSCs are well defined and posses a strict formal semantics [1]. These descriptions are translated in the formally defined Monitor Petri nets [13] to enable a generic code generation for different target platforms. For this reason, simulation and analysis can be performed to verify the correctness of the specification.

(4) In our development process, we use the concept of the Model Driven Development proposed by the OMG[1] for the UML. The transformations are defined by a graphical graph transformation language (Story Driven Modeling) working on repositories generated by MOFLON[2].

(5) Costs can be reduced, because the system developers do not have to learn a special unintuitive language and can use specifications from the early requirements phase through the whole development process by refining them.

(6) In our case, it is important that we support all necessary constructs for behavioral signature modeling. Therefore, Schmerl [14] has evaluated several policy languages and proposed requirements that a general policy language based on Petri nets has to fulfill. He has categorized event patterns with respect to several aspects of the semantics model listed in Table 1. In the next section, we show how these requirements are met by LSCs.

Beside these requirements, the modeling of obligations, permissions, and prohibitions have to be supported. As Soulhaug et al. [18] have stated, these deontic modalities can be expressed by LSC by nature. So obligations can be expressed

[1] Object Management Group: www.omg.org
[2] MOFLON meta-CASE tool: www.moflon.org

Table 1. Requirements for a behavioral signature language

Type of sequences	
Sequence	Several events with a strict sequential order.
Conjunction	Several event patterns that can occur in arbitrary order.
Negation	An event pattern that must not occur.
Disjunction	One event pattern of several is possible.
Simultaneous	Two events that occur at the same time.
Type of Iterations	
Exact	A pattern has to occur n times.
At least	A pattern has to occur at least n times.
At most	A pattern has to occur at most n times.
Continuity	
Continuous	Every event that occurs has to be modeled in the signature.
Non-continuous	The signature is matched if all modeled events have occurred. All additional events between are ignored.
Concurrency	
Non-overlapping	Two or more sequences of events have to occur sequentially. They must not share events during matching.
Overlapping	Two or more sequences of events are allowed to share events during matching .
Context conditions	
Intra-step conditions	A simple boolean expression on an event occurrence.
Inter-step conditions	A complex condition between properties of several events.
Matching rules	
First	The first event matching is bounded.
Last	The last event of several occurrences is bounded.
All	All matching events are bounded.
Type of consumption	
Consuming	In a signature an occurred event is only used for one match in the modeled pattern.
Non-Consuming	In a signature an occurred event is used for several matches in one pattern.

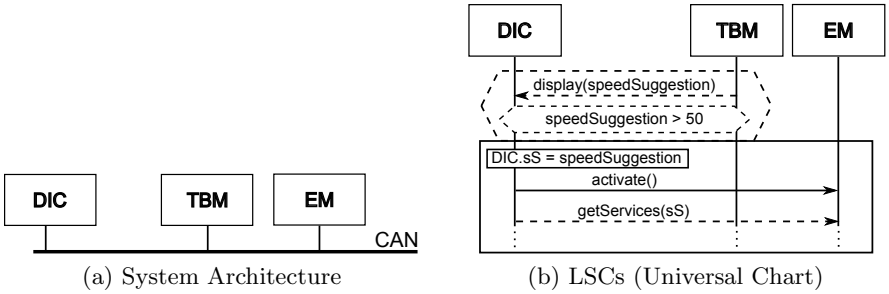
by use cases described by existential charts, permissions by use cases with universal charts (composed of pre- and mainchart), and prohibitions by misuse cases with universal charts.

4 LSCs as Behavioral Signature Modelling Language

In the requirements phase of our development process, Live Sequence Charts are used in combination with structural specifications. Therefore, UML class diagrams describe the structure of the system and specify the participants and their relations. To model the relation between single signatures, UML use cases extended with the concept of misuse cases are used. They declare whether the signature describes an intended behavior (use case) or a faulty behavior or attack (misuse case).

As a detailed specification language for modeling use and misuse cases, we exploit the expressiveness of Live Sequence Charts. After pointing out the requirements that are crucial for a policy language in Sect. 3, we demonstrate by example how LSCs can satisfy these. In the following, we use a scenario, shown in Fig. 2a), based on a CAN bus. Koscher et al. [7] have shown that there are security threats in modern automobiles. Many of them result from the weaknesses of the CAN bus protocol, because packets of this protocol do not include authenticator fields or identifiers for the source. The CAN ID header only contains information about the packet type. Additionally, these packets are broadcast to every node in the network that decides by itself if the packet is relevant. So one compromised component is enough to inject messages on the CAN bus and, thereby, control other nodes of the network. In this way, false information could be displayed on the Driver Information Center (DIC) that is connected to the bus. These corrupted messages are injected by two control units communicating with the outside world. One is a wireless communication module for toll collection (TBM) and the other communicates with some enterprise roadside units (EM).

Before discussing how the requirements raised in Sect. 3 are complied by LSCs, we explain the basic concepts of LSCs by the example in Fig. 2b. In this signature, three instances are involved called *DIC*, *TBM* and *EM*. The vertical lines are lifelines defining a partial order in time from the top to the bottom. The dashed hexagon, a prechart, is special to LSC *universal charts* and represents a precondition that has to be fulfilled before the main chart, the rectangle below, is valid. As in message sequence charts, function calls or messages are modeled by arrows between the lifelines of the instances, conditions



(a) System Architecture (b) LSCs (Universal Chart)

Fig. 2. Car2X Example Scenario

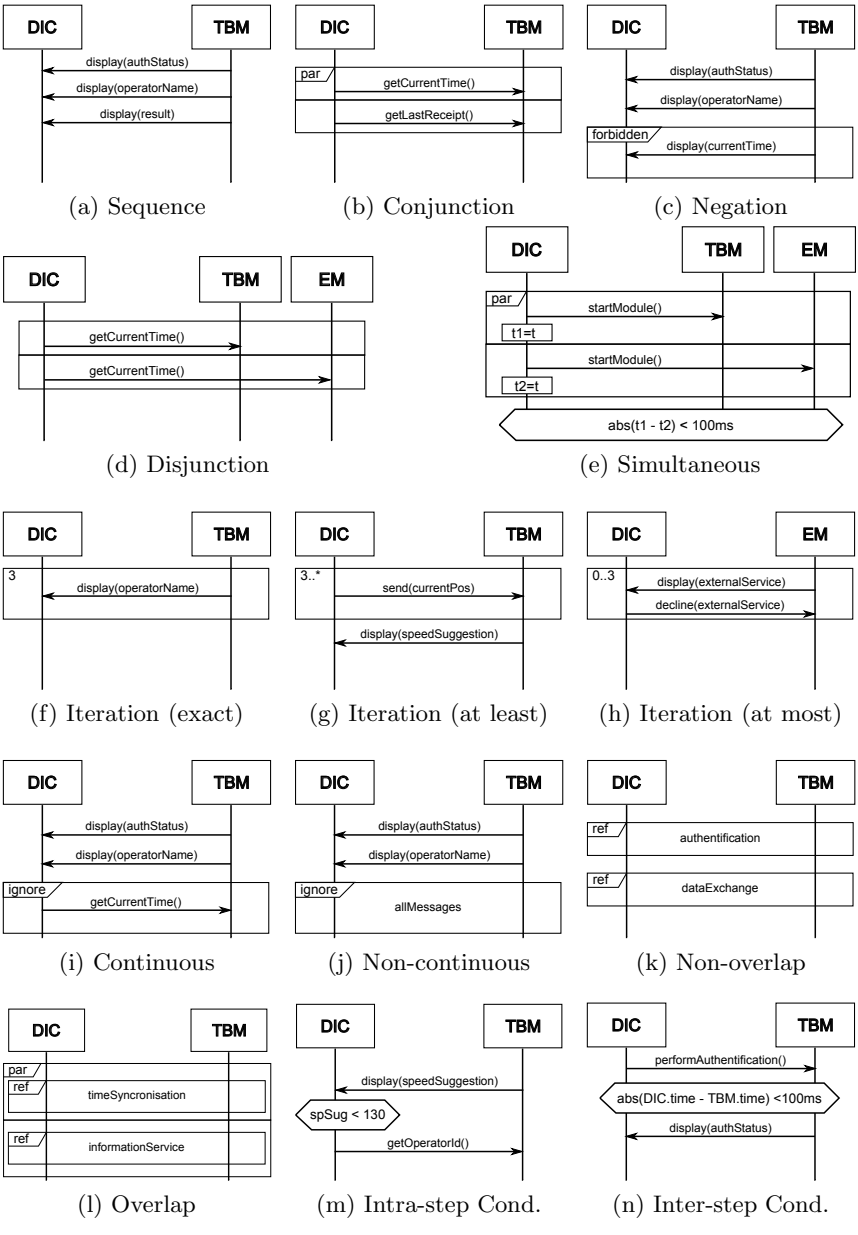
as hexagons and assignments of values in rectangles. Thereby, all dashed lines represent *cold* (optional) and solid lines *hot* (mandatory) elements. The prechart evaluates to true when the *TBM* sends a speed suggestion to the DIC module and the suggested speed is greater than 50. At this speed, the car is considered to be out of town, where additional services are available. Now the main chart has to match. First the speed suggestion is saved in the variable *DIC.sS*. Then the *DIC* has to send an activate message to the *EM* and is allowed to request a service, whereby, the variable *sS* has to be sent as payload. In contrast, signatures that describe obligations can be modeled as an *existential chart*, an LSC without a prechart.

To show how LSCs satisfy the requirements evaluated in the previous section, we will use existential charts for simplicity. In all examples in Fig. 3 single messages can be also considered as complex patterns.

Patterns that describe the type and order of events, are depicted in Fig. 3a) to e). The LSC in a) shows a *sequence* of events that have a fixed partial order on every lifeline and between the sending and receiving of a message. The *TBM* first sends an authentication status to the display of the *DIC*, followed by the operator name of the toll bridge and the result of the negotiation. The next chart describes a *conjunction*, where two messages can be sent in an arbitrary order. To realize this, we have to introduce a construct available for message sequence charts, but missing for LSCs – the *par* fragment. This describes a concurrent occurrence of patterns and will be reused to satisfy further requirements. The property of *negation* is modeled in c) as a *forbidden* fragment. This can be associated with the whole chart as used in the example or limited to a sub chart. So a message to display the time is not allowed during the modeled signature. In Fig. d) a *disjunction*, where only one of the messages is allowed to be sent, is depicted. Therefore, two sub charts are combined to one or-structure. Finally, in subfigure e) the *par* fragment is used to model a *simultaneous* occurrence of two messages. After the messages are sent by *DIC* the time is stored and, afterwards, evaluated by a *hot* condition. Because CAN buses do not allow sending two or more messages at the same time, a time interval that has to be less than 100 ms is tested.

Beside these patterns, a behavioral signature language has to be able to express different kinds of iterations as depicted in Fig. 3f) to h). To model iterations with the properties *exact*, *at least*, and *at most*, standard *loop* fragments of LSCs are used. Thereby, the annotations n, $n..*$, and $n..m$ define the lower and upper bound of repetitions.

When using a signature language, it is important to define how the modeled signatures have to be interpreted. One aspect is the kind of pattern matching that can be *continuous* or *non-continuous*. In the continuous case, all messages occurring in the system during the monitoring of the signature have to be explicitly modeled. Therefore, we have introduced an additional *ignore* fragment that can be used similar to the *forbidden* fragment, but describes messages that are not relevant to the signature and can be ignored. In this way, signatures can be modeled concisely, without allowing every message that is not explicitly modeled as in the non-continuous case. To express that all additional messages



(a) Sequence (b) Conjunction (c) Negation

(d) Disjunction (e) Simultaneous

(f) Iteration (exact) (g) Iteration (at least) (h) Iteration (at most)

(i) Continuous (j) Non-continuous (k) Non-overlap

(l) Overlap (m) Intra-step Cond. (n) Inter-step Cond.

Fig. 3. Required Patterns as Live Sequence charts

should be ignored, an *ignore* fragment can be used, marked with "allMessages". Another aspect is, if two or more patterns can occur concurrently or one after the other. Figure 3k) and l) show how this is realized using the LSC syntax. For *non-overlapping* sequences the patterns are placed in subcharts – here references to other LSCs or for *overlapping* sequences a *par* fragment is used.

The ability to distinguish between *intra-step* and *inter-step* conditions is another essential concept for signature languages. An example for an *intra-step* condition is presented in subfigure m). There, the received suggested speed is checked to be less than 130 before the next message is allowed to occur. A more global condition is the *inter-step* condition, depicted in n), which compares the time on both instances with each other.

The *matching rules* – *first*, *last*, and *all* – can be modeled as single messages or by using loops. To describe a signature that matches *all* occurrences of an event type, can be specified as loop fragment – shown in Fig. 3 f) to h). A single message as depicted in a) is used to match the *first* and a loop followed by a different event type is used to match the *last* occurrence.

For the *type of consumption* we decided to model signatures that are matched in a *consuming* manner, because this fits to the standard LSC syntax. The matching of a signature consumes the occurred events.

As presented in this section, we were able to satisfy all requirements that have been postulated by us and the referred authors in Sect. 3. To accomplish a compact modeling of signatures with LSCs, we had to extend the language by two new concepts, that fit the spirit of LSCs. These are the *par* fragment to describe concurrent patterns and the *ignore* fragment to achieve a compact description of signatures in a continuous pattern matching scenario.

5 Conclusion and Future Work

In this paper, we have shown that Live Sequence Charts already cover most of the crucial properties that a behavioral policy language must possess. Additionally, we have proposed some extensions that are needed to satisfy all the postulated requirements stated in Sect. 3. These extended LSCs are used in our proposed development process for security monitors as an abstract behavioral signature language in the requirements phase. With these extensions, it is possible to model policies in an easy and even to non-practitioners understandable way.

In the future, the LSC descriptions have to be extended by timing constraints, as they are described, e.g., for UML Sequence charts in the MARTE profile. We are currently working on a prototype case tool that supports the whole process from the modeling of policies in the requirements phase until the automatic generation of monitors for different target platforms. It is based on the UML modeling tool Sparx Systems Enterprise Architect that is tailored for the modeling of all diagram types included in our process. Thereby, model-to-model transformations are specified by graph transformations by the meta-modeling tool MOFLON[3].

[3] MOFLON meta-CASE tool: www.moflon.org

Acknowledgements

This work was supported by CASED (http://www.cased.de).

References

1. Damm, W., Harel, D.: LSCs: Breathing Life into Message Sequence Charts. Formal Methods in System Design 19(1), 45–80 (2001)
2. Giarratano, J., Riley, G.: Expert Systems: Principles and Programming, 3rd edn. Course Technology (1998)
3. Groll, A., Ruland, C.: Secure and Authentic Communication on Existing In-Vehicle Networks. In: Proc. of IEEE IV 2009, pp. 1093–1097 (2009)
4. Harel, D., Maoz, S., Segall, I.: Some Results on the Expressive Power and Complexity of LSCs. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds.) Pillars of Computer Science. LNCS, vol. 4800, pp. 351–366. Springer, Heidelberg (2008)
5. Hussein, M., Zulkernine, M.: UMLintr: A UML Profile for Specifying Intrusions. In: 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, ECBS 2006, pp. 8–288. IEEE, Los Alamitos (2006)
6. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 412–425. Springer, Heidelberg (2002)
7. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al.: Experimental Security Analysis of a Modern Automobile. In: 2010 IEEE Symposium on Security and Privacy (SP), pp. 447–462. IEEE, Los Alamitos (2010)
8. Kumar, S.: Classification and Detection of Computer Intrusions. Ph.D. thesis, Purdue University (1995)
9. Lindqvist, U., Porras, P.: Detecting Computer and Network Misuse through the Production-based Expert System Toolset (P-BEST). In: Proceedings of the 1999 IEEE Symposium on Security and Privacy, 1999, pp. 146–161. IEEE, Los Alamitos (2002)
10. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)
11. Massacci, F., Naliuka, K.: Towards Practical Security Monitors of UML Policies for Mobile Applications. In: Proc. of IEEE POLICY 2007, pp. 278 (2007)
12. Papadimitratos, P., Buttyan, L., et al.: Secure Vehicular Communication Systems: Design and Architecture. IEEE Commun. Mag. 46(11), 100–109 (2008)
13. Patzina, L., Patzina, S., Piper, T., Schürr, A.: Monitor Petri Nets for Security Monitoring. In: Proc. of S&D4RCES (2010)
14. Schmerl, S.: Entwurf und Entwicklung einer effizienten Analyseeinheit für Intrusion-Detection-Systeme. Diplomarbeit, Lehrstuhl Rechnernetze, BTU Cottbus (2004)
15. Sindre, G., Opdahl, A.L.: Capturing Security Requirments through Misuse Cases. In: NIK 2001 (2001), http://www.nik.no/2001
16. Sloman, M., Lupu, E.: Security and Management Policy Specification. IEEE Network 16(2), 10–19 (2002)

17. Smith, S., Beaulieu, A., Phillips, W.G.: Modeling Security Protocols Using UML 2. In: Workshop – Modeling Security 2008 (2008)
18. Solhaug, B., Elgesem, D., et al.: Specifying Policies Using UML Sequence Diagrams–An Evaluation Based on a Case Study. In: Proc. of IEEE POLICY 2007, pp. 19–28 (2007)
19. Vigna, G., Eckmann, S., Kemmerer, R.: The STAT Tool Suite. In: Proc. of DISCEX 2000, DARPA Information Survivability Conference and Exposition, 2000, vol. 2, pp. 46–55. IEEE, Los Alamitos (2002)
20. Westphal, B., Toben, T.: The Good, the Bad and the Ugly: Well-Formedness of Live Sequence Charts. In: Baresi, L., Heckel, R. (eds.) FASE 2006. LNCS, vol. 3922, pp. 230–246. Springer, Heidelberg (2006)

Detecting Illegal System Calls Using a Data-Oriented Detection Model

Jonathan-Christofer Demay[1], Frédéric Majorczyk[2],
Eric Totel[1], and Frédéric Tronel[1]

[1] Supelec, Rennes, France
first_name.last_name@supelec.fr
[2] IRISA / Université de Rennes 1, Rennes, France
first_name.last_name@irisa.fr

Abstract. The most common anomaly detection mechanisms at application level consist in detecting a deviation of the control-flow of a program. A popular method to detect such anomaly is the use of application sequences of system calls. However, such methods do not detect mimicry attacks or attacks against the integrity of the system call parameters. To enhance such detection mechanisms, we propose an approach to detect in the application the corruption of data items that have an influence on the system calls. This approach consists in building automatically a data-oriented behaviour model of an application by static analysis of its source code. The proposed approach is illustrated on various examples, and an injection method is experimented to obtain an approximation of the detection coverage of the generated mechanisms.

1 Introduction

Generally speaking, an attack against an application consists in exploiting a vulnerability in order to violate the confidentiality or integrity properties of the system or the application under attack. In the context of intrusion detection methods at application level, a lot of existing work focuses on the detection of the violation of the integrity property. Attacks against a process can consist in corrupting either the control-flow of the program (e.g., to execute injected code), or the data items manipulated by the program during its execution. A lot of papers focus on the detection of the program control-flow corruption, either considering the process as a white box, or seeing it as a black box. An exemple of a white box approach is to verify during the execution that the control-flow graph of the program is legal. An example of the black box approach consists in verifying that the trace of the process execution in the system is correct (e.g., the sequence of system calls [1]). Both approaches can be subject to false negatives, as the attacker can either corrupt data items that do not influence the control-flow of the program, or perform attacks that mimic [2] the normal behaviour of the application. Various papers [3,4,5,6] have enhanced the black box approach in order to detect these types of refined attacks.

In this paper, we propose a white box approach for intrusion detection that aims at detecting the corruption of the data items in an application, so as to detect erroneous system calls (e.g., their arguments are not correct, or the data that led to their execution

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 305–316, 2011.

were incorrect). The approach relies on the building of a data-oriented behaviour model. This method can be presented as an interesting complement to the usual control-flow corruption detection method, in order to detect data oriented attacks. To attain this goal we use static analysis to build constraints on intrusion sensitive data items, then we instrument the software with executable assertions that check these constraints during the execution of the program.

The contribution of this paper is not to provide new static analysis techniques, as our work relies on an off-the-shelf static analyser called Frama-C [7]. However, we want to show on real-life examples that a detection model can be built by static analysis and detect data attacks (even unknown ones).

The paper is organized in the following way: after a short related work section on white-box attack detection, we show how to build the behaviour model and emphasize the accuracy of the model on a previously known attack. Then we show the results of the software instrumentation on various examples. At the end we evaluate on an example the detection rate we can expect from the generated detection mechanisms.

2 Related Work

We believe that white box mechanisms can help improving the detection performance as they are able to take advantage of the internal state of the monitored program. Indeed, they have access to all the internal data structures and algorithms used by the program.

That is the case, for example, with Control-Flow Integrity [8] and Program Shepherding [9]. These generic techniques verify the integrity of the control-flow of a program. A control-flow graph of the program is computed prior to its execution and then used at run time to check the integrity of the process control-flow. Because mimicry attacks still need to force the program control-flow to deviate from valid execution paths, they are caught by these approaches. However, unlike our approach, all those techniques are completely ineffective against computation data attacks (also called non-control data attacks [10]), since these attacks are performed using a valid execution path.

Other white box approaches that focus on non-control-data attacks and that do not exhibit this weakness have been proposed. For example, Write-Integrity Testing [11] enforces control-flow and data-flow integrity in a program. In the work on Data Flow Integrity [12], a data-flow graph is computed prior to the execution. It contains, for each data item read by an instruction, the set of instructions that may have written its current value. This data-flow graph is then used at run time to verify the integrity of the data flow of the process. If the program has a vulnerability that is exploited to corrupt some data, the next time this data is read a deviation from the data-flow graph will be observed allowing thus the detection of the attack. This type of approach is very effective against all kinds of non-control-data attacks, but use a very different philosophy than our approach. They focus on the illegal modification of the data, whereas in our approach we focus on the correctness of the data. As a consequence some attacks missed by the data-flow integrity method (such as an illegal value stored in a correct variable) can be detected by our approach. Conversely some illegal writes can be missed by our approach (a legal value can be written in an incorrect variable), making both approaches complementary.

3 Intrusion Detection

In this section, we explain how non-control-data attacks are real threats and how a data-oriented behavior model can detect them. We also present *SIDAN* [1] (Software Instrumentation for the Detection of Attacks on Non-control-data) [13], a tool we have developed that implements our detection model.

3.1 An Attack against Non-Control-Data

Chen et al. [10] have demonstrated that non-control data attacks can be as severe as control-data attacks on various real world vulnerabilities. Among them, a vulnerability found in the implementation of the open source ftp server *wu_ftpd* will serve as an example to illustrate our approach. Figure 1 (left column) is an excerpt of the original code exhibiting the same vulnerability. Line 10, a string taken as user input (line 7) is printed without using a string format. Consequently, a user can forge an incorrect buffer containing string formats that allows to write directly in memory. In this case, the target could be the *uid* variable. As a consequence, the attacker can elevate its privilege at line 12, without corrupting the execution path, by forcing the parameter of the *seteuid* call to be the administrator identifier (zero). This example shows how such an attack violates a very simple constraint on the *uid* variable. Indeed, the *uid* variable should remain constant during the execution of the loop (lines 7 to 13) and should be equal to the value it has been assigned at *uid* (line 2). The problem we tackle in this paper is to automatically build such constraints in order to detect attacks at runtime.

```
00. int main(int argc, char ** argv){
01.     char buffer[256];
02.     uid_t uid = 5;
03.
04.
05.     seteuid(uid);
06.
07.     while(aux = fgets(buffer, 256, stdin))
08.     {
09.         seteuid(0);
10.         printf(buffer);
11.
12.         seteuid(uid);
13.     }
14. }
```

```
00. int main(int argc, char ** argv){
01.     char buffer[256];
02.     uid_t uid = 5;
03.
04.     assert(uid == 5);
05.     seteuid(uid);
06.
07.     while(aux = fgets(buffer, 256, stdin))
08.     {
09.         seteuid(0);
10.         printf(buffer);
11.         assert(uid == 5);
12.         seteuid(uid);
13.     }
14. }
```

Fig. 1. Example of string format vulnerability and useful assertions

3.2 Data Oriented Detection Model

In our approach, we consider that an attacker aims at modifying data items in the memory space of a process in order to execute one or more incorrect system calls. This objective can be fulfilled in two ways: either the attacker alters variables that influence the internal control-flow of the program (and thus executes system calls in an incorrect context), or the attacker modifies directly or indirectly the values of the parameters of one or more system calls (and thus executes legal system calls with incorrect values).

[1] http://www.rennes.supelec.fr/ren/rd/ssir/outils/sidan/

Both types of attacks aims at modifying non-control data items in the program. Note here that non-control data items are all variables used by the program source code, and can thus have an impact on the control-flow of the application. They are opposed to control-data items that are used by the system (and not the application) to control the execution flow of the application (e.g., a return adress on the stack).

In order to detect these modifications, we propose to identify the set of constraints that should be verified at runtime for these items. Generally speaking, these constraints can be divided in two classes: the variation domain of the variables (e.g., a variable can take a restricted set of values), and the relationship between the variation domains of the variables (i.e., when a variable has particular values, other variables take a defined set of values). If we only check if a variable is within its variation domain, it may be easy for an attacker to impose a reasonable value that would fit in the variation domain, but that is incorrect in the context of the program. Clearly, if we can maintain the relationship with other variables, it will be more difficult for an attacker to modify simultaneously several variables that depend on each other while keeping the program in a consistent state. As a consequence, we propose to define a data behaviour model for intrusion detection that aims at taking into account these requirements.

Formally, we define for a given system call SC_i its data behavior model by a triple (SC_i, V_i, C_i) where V_i is the set of variables the system call depends on, and C_i the set of constraints on these variables that can be deduced from the program analysis. We can define the normal data behavior model of the program by the set of all triples, $DBM = \{\forall i, (SC_i, V_i, C_i)\}$. In the following section, we address the two problems faced to build this model: how to determine the set of variables a system call depends on, and how to obtain the constraints that must be verified on these variables at runtime.

Building the set of variables. Building V_i requires the ability to determine in the program which are the variables that influence the execution of the particular system call SC_i. Generally speaking, a system call can depend on a variable in two different ways: a variable either has an influence on the path in the program that leads to the execution of SC_i or influences the parameters of SC_i. These sets of variables can be built by using a static analysis technique called program slicing [14]. A program slice can be defined as the parts of a program that potentially affect the values computed at some point of interest of this program. In our case, we are looking for all the variables that influence a system call, and thus all variables that are in the program slice whose point of interest is the system call itself. In the static analysis field, the computation of a program slice is generally based on the computation of a program dependency graph (PDG) [15]. The PDG is a directed graph whose vertices correspond to statements and control predicates, and edges correspond to data and control dependencies. This graph can be used to exhibit the set of variables a particular system call depends on, and the type of dependency. In our implementation, we directly use the PDG notion to discover in the program all variables a system call depends on. To illustrate this paragraph, we can consider the example on Figure 1: the *seteuid* call at line 5 depends on one variable: *uid*. However, the *seteuid* call at line 12 depends indirectly on the *aux* variable and directly on the *uid* variable.

Constraint discovery. Automatically discovering constraints in the source code on the variables that are defined in the previous paragraph requires to use static analysis

techniques. We could imagine any types of constraints, including for example temporal constraints. In practice, static analysis techniques often compute constant constraints, also called invariants. Indeed, any static analysis technique that is able to compute invariants from the source code fits our needs. Moreover, a popular technique for calculating such invariants is the abstract interpretation method [16]. In practice, abstract interpretation provides a way to find properties on the variables of a program by computing abstract domains that represent abstractions of the real properties of the program. Several models have been developed to discover such invariants. Among them, we have chosen to focus on the build of numerical abstract domains, i.e., we intend to find numerical invariants. These types of domains can be classified in two groups: non-relational domains that find numerical properties on variables individually, and relational domains that permit to find numerical properties on logically linked variables. Non-relational domains include for example the interval domain [16] (wich permits to find invariants of the form $v_i \in [c_1, c_2]$ where v_i is a variable of the program and c_1 and c_2 are numerical constants), the constant propagation domain ($v_i = c$) and the congruence domain [17] ($v_i \in a\mathbb{Z} + b$). Example of relational domains can be cited such as the polyhedron domain [18] ($\alpha_1 v_1 + ... + \alpha_n v_n \leq c$), the linear equality domain [19] ($\alpha_1 v_1 + ... + \alpha_n v_n = c$) and the linear congruence equality domain [20] ($\alpha_1 v_1 + ... + \alpha_n v_n \equiv a[b]$). The problem with relational domains is that the algorithms they use usually do not scale on large programs. That is why *Frama-C* uses computational methods that are based on non-relational domains.

SIDAN Plugin in the Frama-C framework. We implemented in *SIDAN* the computation of numerical constraints for a given system call. *Frama-C* provides a *Value Analysis* plugin that is able to provide a computation of the variation domains of the variables that influence the function calls. This plugin provides constraints of the type *"integer variable x lies within the domain* [0,5] *in all executions"* as a result. If we consider the example Figure 2, the assertion generated for the call to the function f, using the *Value Analysis* plugin of *Frama-C* alone would be $a \in \{1,2\}$ and $b \in \{0,1\}$. Indeed, as the Value Analysis plugin uses a non-relational abstract domain, his result misses the relation between the variables a and b.

If we consider the program Figure 2, we see that when $b == 0$ then $a == 1$, and when $b == 1$ then $a == 2$. Actually, to obtain this result we have to consider that there are two paths leading to the call to the function f, and that the constraint to verify at the call to f should take these two paths into account. The *Value Analysis* plug-in uses an algorithm that can potentially keep in memory several invariants computation on

```
00: extern int a, b;
01: void f(int);

03: void g(){
04:   if (b == 0) a = 1;
05:   else if(b == 1) a = 2;
06:   else return;


09:   f(a);
10: }
```

```
00: extern int a, b;
01: void f(int);

03: void g(){
04:   if (b == 0) a = 1;
05:   else if(b == 1) a = 2;
06:   else return;

08:   assert((a == 1 && b == 0) || (a == 2 && b == 1))
09:   f(a);
10: }
```

Fig. 2. *C* code sample that emphasises relations between variables

several execution paths. The plug-in can be parametrized to define the number of paths explored in parallel by the *Value Analysis* plug-in, which is related to the number of states it keeps in memory before computing an union. If the number of paths explored in parallel is sufficient, the *Value Analysis* plug-in now has internally the information required to build these kinds of constraints. By using a hook in the *Value Analysis* plug-in, it is possible for our plug-in to access this internal information while the analysis is performed. Thus, it allows us to build the invariant by using the variation domain of all the variables on each path. In the example we have described, the invariant generated for line 08 is $(((b == 0) \wedge (a == 1)) \vee ((b == 1) \wedge (a == 2)))$.

Note that the example we give here focuses on invariants computed for integers. In practice, the *Value Analysis* plug-in performs well on integers and floats, but is not very efficient for pointer analysis (at best, it detects access to an unallocated buffer and some out-of-bound access). Discovering constraints on strings is also unavailable due to the fact that the specification of the standard string functions is not included in *Frama-C*. In order to build some constraints on strings, we have preprocessed the source code to replace standard string comparisons by a set of character comparisons whenever possible (see Figure 3 line 3). As a result, some constraints on string buffers have been obtained in the programs we tested our approach on.

3.3 Generated Assertions

In order to verify the constraints in the program, we insert executable assertions (see Figure 2 line 08), which is a technique heavily used in the dependability domain, and more precisely in defensive programming [21,22].

The constraints that we can compute for a given system call deal with the variables that are available locally in the context of the system call. However, this call generally depends not only on the local variables but also on the variables manipulated by previous functions in the call stack. That is why it is necessary to compute invariants for all function calls that are on the path that leads to the system call. This implies that we must distribute the executable assertions on all the paths that lead to system calls. Moreover, some system calls can be performed in functions located in external libraries. As a consequence, we choose to insert executable assertions in front of each function call.

To demonstrate the assertion generation capabilities of our data-oriented detection model, we first use as an example a vulnerable version of *OpenSSH*.

The code in Figure 3 is inspired by this vulnerable version of *OpenSSH* and reproduces the basic structure of the real code. The vulnerability is located in the *packet_read* function and can be used to overwrite the value of the *passwd* variable with an empty string during the execution of *do_authloop*. This allows a successful authentication on the system with any known account (e.g., *root*) and without having to provide a valid password.

Among the assertions generated, the one located at line 11 in the example in Figure 3 has been produced by our plug-in and detects this attack against the program state.

In order to figure out the capability of our tool to generate assertions on common programs, we have applied it on SSH servers (*OpenSSH* and *Dropbear SSH*), http servers (*fnord* and *ihttpd*), and a smtp server (*ssmtp*). The results are summarized in Table 1. As a result, we could say that the number of assertions generated is obviously heavily dependant on the program source code.

```
00: void do_authentication(){
01:   int auth = 0;
      ...
03:   if(!strcmp(pwd, ""))
        /* for users with no password */
05:   else
        /* do_authloop(); */
07:     while(auth != 1) {
08:       type = packet_read(data);
09:       switch (type) {
10:         case SSH_CMSG_AUTH_PASSWORD:
11:
12:           auth = auth_password(pwd, data);
13:           break;
14:         ...
15:       }
16:     }
17:   do_authenticated(user);
18: }
```

```
00: void do_authentication(){
01:   int auth = 0;
      ...
03:   if(pwd[0] != '\0')
        /* for users with no password */
05:   else
        /* do_authloop(); */
07:     while(auth != 1) {
08:       type = packet_read(data);
09:       switch (type) {
10:         case SSH_CMSG_AUTH_PASSWORD:
11:           assert(pwd[0] != '\0');
12:           auth = auth_password(pwd, data);
13:           break;
              ...
15:       }
16:     }
17:   do_authenticated(user);
18: }
```

Fig. 3. Example inspired from *OpenSSH*

Table 1. Assertions generated

	OpenSSH	DropbearSSH	ihttpd	fnord	ssmtp
Number of lines	38000	11000	1043	2303	2976
Number of assertions	291	91	145	41	240
Computation time	6 hours	3 hours 45 minutes	1 minute 17 seconds	5 hours	22 minutes

4 Assessment of the Detection Mechanisms

Even though it is possible to test our detection mechanism against various real world attacks such as those described in [10], such a method would only cover a very small subset of all possible attacks. In order to evaluate the detection coverage of our approach, we would need to know all the vulnerabilities that afflict a program as well as every possible way of exploiting them. As it is not possible to automatically compute this from the source code, we need to define another method to evaluate the detection coverage of our model. In this section we propose a method to assess the detection mechanisms by simulating attacks against non-control-data items without prior knowledge of the vulnerabilities. Our goal is to simulate the consequences of non-control-data attacks by directly modifying in the process memory space the data items it is currently manipulating. In this section, we propose an approach to evaluate our detection mechanism that is similar to the ones proposed in the security field to help discover new vulnerabilities (fuzzing) and in the dependability field to evaluate fault detection and tolerance mechanisms (fault injection).

4.1 Simulation of Attacks against Non-Control Data

Generally speaking, a particular vulnerability usually allows the attacker to access a limited part of a process memory. However, in the worst case scenario it can give to an attacker an access to the whole memory space of a process. For that reason, our injection mechanism is given access to potentially every internal data item of the program under

test. However, to accurately simulate a real non-control-data attack, we want to restrict (1) the locations and (2) the instants where an injection can occur during the execution of a program. Firstly, during such an attack not every data item is a potential target. The data items that may be of interest for an attacker are within the subset of data items that can influence the execution of the system calls. Consequently, we target only these data items (they define the locations of potential injections). Other data items are irrelevant for our simulation approach. Secondly, we will modify such items only when they are currently in use (i.e., when they are influencing the current execution of the program).

4.2 Code Instrumentation and Fault Injection

To simulate this injection model, two problems have to be addressed: how do we determine the set of data items that are potential targets for a non-control-data attack, and how do we determine for each one of them when it is appropriate to inject a corrupted value. Clearly, the set of data we want to modify is the very same set of data items we have defined in Section 3.2.

The simplest way to determine the memory address of a variable we want to inject is to obtain it at execution time. This is why we have choosen to also embed the corrupting mechanisms within the source code. Moreover we have decided to distribute the injection mechanisms when the corresponding variables are reachable, that is right before every function call that depends on them. We used the same approach as described in Section 3.3 where we discussed the distribution of the detection mechanisms. In the end, each candidate function call is preceded by a call to the corrupting function implemented by a single external function called *inject()*.

Each injection point is assigned a unique identifier. This identifier is passed as a parameter to the injection function. The remaining arguments are the number of variables that can be corrupted and for each one of them, its address and its size. The corrupting function is controlled by an external process using environment variables. This process controls the unique identifier of the injection that is to be activated, the variable that will be corrupted and the value used to perform the injection. An injection is triggered only once, even when the call to the corrupting function happens many times (*e.g.*, in a loop). The tool presented in Section 3 has been modified in order to perform the instrumentation needed by our injection mechanism. Note that the set of variables A used in an assertion is always a subset of the set of variables I used in the injection process (see Figure 4). Indeed, the injection can be performed in any variable that influences the function call, unlike the assertions that only concern variables for which value constraints have been discovered.

Our goal is to evaluate our detection mechanism presented in Section 3. To do that, we need to cover a large set of memory corruptions that might be used by a malicious user to perform an intrusion. Very much like a fuzzing technique, we are going to randomly put the internal state of the process in an erroneous state. We perform various injections during the execution of the program used in our test environment in order to simulate the result of a vulnerability exploitation.

To activate a maximum of function calls in the program, we have written a set of scenarios whose goal is to make the control-flow pass through a maximum number of function calls. In the case of *Dropbear SSH*, we have written a set of 24 scenarios that allows us to reach 92% of the function calls.

```
extern int a;
const int b = 1;

if (a == 0) {
  inject(0,2,&a,sizeof(a),&b,sizeof(b));
  assert(b == 1 && a == 0);
  f(b);
}
```

$A \equiv I$

```
extern int a;
extern int b;

if (a) {
  inject(0,2,&a,sizeof(a),&b,sizeof(b));
  assert(a != 0);
  f(b);
}
```

$A \subset I$

```
extern int a;
extern int b;

if (a == b) {
  inject(0,2,&a,sizeof(a),&b,sizeof(b));

  f(b);
}
```

$A \equiv \emptyset$

Fig. 4. Different cases of injections and assertions

During the injection process, for each function call that can be reached by a scenario, a random variable from the set of variables that influences the execution of this function call is chosen to be injected with a random value. Each time an attack is simulated, the controller logs if the scenario ended properly or if the process exited unexpectedly or found itself in a deadlock and needed to be killed after a time-out. The controller also logs the behavior of the process during the attack (in terms of system calls and their arguments). The whole test setup is shown in Figure 5.

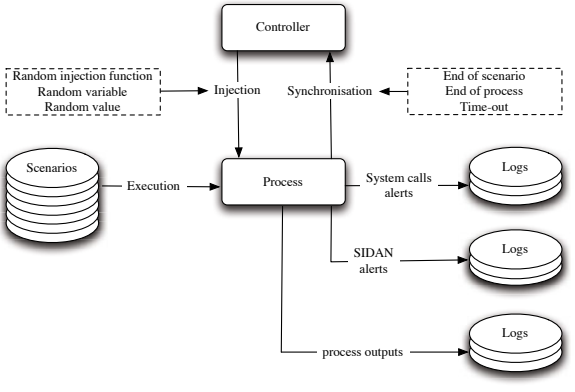


Fig. 5. Experimentation protocol

4.3 Evaluation Results

Using the experimentation protocol described in Section 4.2, we have performed a total of 120 000 injections on the *Dropbear SSH* server. As explained before for each injection, we have logged three kinds of information. Firstly, we have compared the output generated by the server during the injection with respect to the output generated without injection. These observations can be considered as an extremely accurate indicator of a potential attack. Indeed, in these cases, the modification of a single variable has been able to modify the execution of the SSH server upto the point its external behaviour (as seen by an SSH client) was changed. Note that 69.36% of the injections have lead to such an alert (either a deviation of output, or a crash of the server). Of course, while being an extremely accurate way of detecting intrusions, this approach is difficult to generalize in real life settings, since it would require to compare the output

produced by the server for each command it receives with a reference output. Considering the generally extremely large set of outputs such a server can produce, this approach is hopeless. Here we were able to use such an approach because of the limited set of scenarios we have used during the assessment. Secondly, we also recorded the set of system calls (with their arguments) that were generated during normal executions of the different scenarios (training), and during injections. These recordings have been submitted afterward to an offline intrusion detection mechanism [3]. Once again, this IDS was settled in optimum conditions, since it was trained for a given scenario. And even in these optimal conditions, note that it only detects 22.48% of injections.

Finally we have recorded the alerts generated by our SIDAN tool. The results of all these measures are summarized in Table 2. A more detailed version of the obtained results is given by the Figure 6. We can see that SIDAN detects 62.36 % of the injections. This detection rate is comparable to the one obtained by the first IDS based on the comparison of the output generated by the server (but recall here, that we claim that this kind of IDS is extremely difficult to build in real settings). However, SIDAN is still prone to false negatives with at most 37.64% of injections missed. We can refine these figures by taking into account the fact that within these 37.64% of cases where SIDAN raised no alert, 10.63% where cases where : (1) neither the output generated by the SSH server deviated from the reference output. (2) nor the system call trace deviated from the reference trace. We can be highly confident that these cases do not correspond to exploitable attacks. Hence we can subtract these 10.63% from the figures obtained for false negatives for SIDAN. All in all, we can claim that the rate of false negatives for SIDAN lies within a 27.01% and 37.64%.

Table 2. Injection results on *Dropbear SSH*

	SIDAN alert	Unexpected server exit	Incorrect server output	Strace alert
Injection detected	74827	21574	61470	26970
Detection rate	62.36%	18.13%	51.23%	22.48%

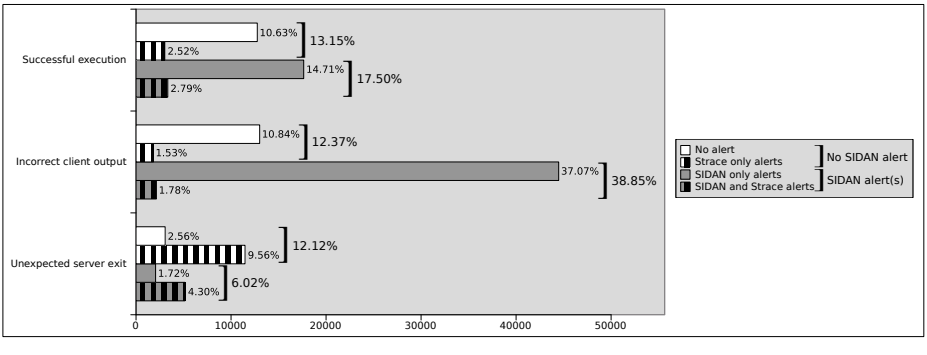


Fig. 6. Distribution of alerts

5 Conclusion and Future Work

In this article, we propose a sofware-level intrusion detection approach based on the internal state of the process that detects data attacks, which are missed by traditional control-flow approaches. Our mechanism relies on a data-oriented behavior model to detect erroneous states that could lead to illegal system calls. We present a tool that implements our approach by analyzing and instrumenting a program's source code. This tool has proved that our approach is useable in the context of real software and that it can detect real world non-control-data attacks (such as the null password attack on *OpenSSH*). We also propose a method to assess these intrusion detection systems against data attacks by using a fault injection mechanism. In the particular case of *Dropbear SSH*, by using our evaluation method, we have estimated, without prior knowledge of any attacks, an approximation of the detection coverage of our detection model.

However, the current implementation of our tool computes the constraints needed by our detection model using only variation domains. This is clearly a limitation, because it does not permit the detection of data attacks on variables whose variation domain is statically unknown in the source code. That is why in the future we intend to use additional static analysis techniques to discover more constraints. We also plan to investigate for our evaluation method the possibility of replacing the set of hand written scenarios by automatically generated scenarios using fuzzing techniques [23].

References

1. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security (1998)
2. Kruegel, C., Kirda, E., Mutz, D., Robertson, W.: Automating mimicry attacks using static binary analysis. In: 14th Conference on USENIX Security Symposium (2005)
3. Kruegel, C., Mutz, D., Valeur, F., Vigna, G.: On the detection of anomalous system call arguments. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 326–343. Springer, Heidelberg (2003)
4. Bhatkar, S., Chaturvedi, A., Sekar, R.: Dataflow anomaly detection. In: 2006 IEEE Symposium on Security and Privacy (S&P 2006) (2006)
5. Mutz, D., Robertson, W., Vigna, G., Kemmerer, R.: Exploiting execution context for the detection of anomalous system calls. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 1–20. Springer, Heidelberg (2007)
6. Feng, H., Kolesnikov, O., Fogla, P., Lee, W., Gong, W.: Anomaly detection using call stack information. In: IEEE Symposium on Security and Privacy, p. 65 (2003)
7. CEA: Frama-c, framework for modular analysis of c
8. Abadi, M., Budiu, M., Erlingsson, U., Ligatti, J.: Control-flow integrity. In: CCS 2005: Proceedings of the 12th ACM Conference on Computer and Communications Security (2005)
9. Kiriansky, V., Bruening, D., Amarasinghe, S.: Secure execution via program shepherding. In: Proceedings of the Usenix Security Symposium (2002)
10. Chen, S., Xu, J., Sezer, E., Gauriar, P., Iyer, R.: Non-control-data attacks are realistic threats. In: Usenix Security Symposium (2005)
11. Akritidis, P., Cadar, C., Raiciu, C., Costa, M., Castro, M.: Preventing memory error exploits with *wit*. In: 2008 IEEE Symposium on Security and Privacy (2008)

12. Castro, M., Costa, M., Harris, T.: Securing software by enforcing data-flow integrity. In: 7th USENIX Symposium on Operating Systems Design and Implementation (2006)
13. Demay, J.C., Totel, E., Tronel, F.: Sidan: a tool dedicated to software instrumentation for detecting attacks on non-control-data. In: 4th International Conference on Risks and Security of Internet and Systems (CRISIS 2009), Toulouse (October 2009)
14. Weiser, M.: Program slicing. IEEE Transactions on Software Engineering (1982)
15. Kuck, D.J., Kuhn, R.H., Padua, D.A., Leasure, B., Wolfe, M.: Dependence graphs and its use in optimization. In: 8th ACM Symposium on Principles of Programming Languages (1981)
16. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (1977)
17. Granger, P.: Static analysis of arithmetical congruences. International Journal of Computer Mathematics 30, 165–190 (1989)
18. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (1978)
19. Karr, M.: Affine relationships among variables of a program. Acta Informatica, 133–151 (1976)
20. Granger, P.: Static analysis of linear congruence equalities among variables of a program. In: TAPSOFT 1991, pp. 169–192 (1991)
21. Goloubeva, O., Rebaudengo, M., Reorda, M.S., Violante, M.: Soft-error detection using control flow assertions. In: Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2003) (2003)
22. Vemu, R., Abraham, J.A.: Ceda: Control-flow error detection through assertions. In: Proceedings of the 12th IEEE International On-Line Testing Symposium (2006)
23. Neves, N., Antunes, J., Correia, M., Verissimo, P., Neves, R.: Using attack injection to discover new vulnerabilities. In: Conference on Dependable Systems and Networks (2006)

IFIP Technical Committee 11 Security and Privacy Protection in Information Processing Systems*

Kai Rannenberg, SH (Basie) von Solms, and Leon Strous

Abstract. IFIP Technical Committee 11 (TC-11) on Security and Privacy Protection in Information Processing Systems was created in 1983 under the chairship of the late Kristian Beckman of Sweden. Representatives from more than 30 IFIP member societies are members of TC-11 and meet at least once a year at the IFIP/Sec conferences that are held in different member countries.

This text gives an overview on the state of TC-11 and its development over the last 28 years. It starts with a snapshot on the current situation of TC-11, followed in Section 2 by an overview of the historical background and trends of TC-11 and its flagship conference IFIP/Sec. Section 3 is dedicated to the main development trends in the field of TC-11, while Section 4 honours the awardees of TC-11's Kristian Beckman Award, many of them TC-11 Pioneers. Section 5 then gives an outlook on the future role of TC-11.

Keywords: Security, Privacy, Protection, Information Processing Systems, IT Systems, ICT Systems, IFIP, IFIP Technical Committee 11.

1 TC-11 – A Snapshot

TC-11 can to some degree be recognized by its aims, scope, and last but not least its working groups, which are introduced in the following sections. All three underwent some significant changes over the past 28 years, which are documented in the remainder of this article.

1.1 TC-11 Aims

To increase the trustworthiness and general confidence in information processing and to act as a forum for security and privacy protection experts and others professionally active in the field.

---

* This text is an updated version of "IFIP Technical Committee 11 Security and Privacy Protection in Information Processing Systems", pp. 302–310 in: Kai Rannenberg, Vijay Varadharajan, Christian Weber (Eds.): "Security and Privacy - Silver Linings in the Cloud", Proceedings of the 25th IFIP TC 11 International Information Security Conference, SEC 2010, held as part of WCC 2010, Brisbane, Australia, September 20-23, 2010, IFIP Advances in Information and Communication Technology, Vol. 330, 2010, ISBN 978-3-642-15256-6, and pp. 153–161 in Klaus Brunnstein and Heinz Zemanek (Eds.): "50 years of IFIP - Development and Visions", ISBN: 978-3-901882-43-2, www.ifip.org.

J. Camenisch et al. (Eds.): SEC 2011, IFIP AICT 354, pp. 317–325, 2011.

1.2 TC-11 Scope

Work towards:

- the establishment of a common frame of reference for security and privacy protection in organizations, professions and the public domain;
- the exchange of practical experience;
- the dissemination of information on and the evaluation of current and future protective techniques;
- the promotion of security and privacy protection as essential elements of information processing systems;
- the clarification of the relation between security and privacy protection.

1.3 TC-11 Working Groups

Already in 1985 TC-11 established its first working groups. Since then number and activity of TC-11's WGs underwent a non-linear but steady growth with two new WGs being established in 2010 driving the number of TC-11s WGs up to twelve, of which two WGs are shared with fellow TCs. The current WG list reads as follows:

1. WG 11.1: Information Security Management (established 1985)
2. WG 11.2: Pervasive Systems Security (established 1985 as the WG on Office Automation and from 1992 until 2009 named Small System Security)
3. WG 11.3: Data and Application Security (established 1987 under the name of Database Security and renamed 2001)
4. WG 11.4: Network & Distributed Systems Security (established 1985 under the name of Crypto Management and from 1992 until 2003 named Network Security)
5. WG 11.6: Identity Management (established 2006)
6. WG 9.6 / 11.7: IT Misuse and The Law (established 1990)
7. WG 11.8: Information Security Education (established 1991)
8. WG 11.9: Digital Forensics (established 2004)
9. WG 11.10: Critical Infrastructure Protection (established 2006)
10. WG 11.11: Trust Management (established 2006)
11. WG 11.12: Human Aspects of Information Security and Assurance (established 2010)
12. WG 8.11 / 11.13: Information Systems Security Research (established 2010)

2 The Historical Background of TC-11 and Its Flagship Conference

In May 1983, the 1st International Conference on Information Security, IFIP/Sec '83, took place in Stockholm, Sweden. This conference was organized by members of the Swedish Special Interest Group on Information Security, as well as a number of further people, including some from existing IFIP Committees. The organization was under the chairman-ship of Kristian Beckman of Sweden. A proposal was submitted to IFIP's General Assembly (GA), and at its meeting in September 1983 in Paris, TC-11 was formally established. Kristian Beckman was appointed as the first Chairman of TC-11.

The 2nd International Conference on Information Security, IFIP/Sec '84, took place in May 1984 in Toronto with the motto "Computer security: a global challenge". During this conference, the first official meeting of TC-11 was held. Unfortunately, because of ill health, Kristian Beckman could not attend that meeting. He asked Per Hoving from Sweden to act as Chairman, but during the conference, the sad news that Kristian Beckman passed away, reached TC-11.

The next TC-11 meeting took place in Dublin during IFIP/Sec 85, which had the motto "Computer security: The practical issues in a troubled world". Per Hoving was elected as Chairman for a three year term, with Willis Ware from the USA as Vice-Chairman.

Subsequent IFIP/Sec Conferences took place as follows and show the real global approach of TC-11 both with regards to its flagship conference as well as its management teams:

- IFIP/Sec 86 Monte Carlo: "Security and protection in information systems"
- 1987: No IFIP/Sec Conference took place, but a TC-11 meeting was held in Vienna (Austria) in conjunction with a TC-11 Working Group conference.
- IFIP/Sec 88 Gold Coast, Australia: "Computer security in the age of information". At the corresponding TC-11 meeting Bill Caelli from Australia was elected as new Chair, Willis Ware re-elected as Vice-Chair, and David Lindsay from the UK as Secretary.
- 1989: No IFIP/Sec conference was held, and efforts were combined with the IFIP Congress which took place in San Francisco, USA.
- IFIP/Sec 90 Helsinki, Finland: "Computer security and information integrity"
- IFIP/Sec 91 Brighton, England: "Information security"
- IFIP/Sec 92 Singapore: "IT security: the need for international cooperation"
- IFIP/Sec 93 Toronto, Canada: "Computer security". Unfortunately David Lindsay died before IFIP/Sec 93. Bertil Fortrie from the Netherlands took over as Secretary.
- IFIP/Sec 94 Curaçao: At the TC-11 meeting during this conference Sebastiaan von Solms from South Africa was elected as Vice-Chair and David Bachelor from Canada as Secretary. Later that year Sebastiaan von Solms was appointed as acting Chair of TC-11 by the IFIP President.
- IFIP/Sec 95 Cape Town, South Africa: "Information security - the next decade". At the TC-11 meeting preceding the conference Sebastiaan von Solms was elected as new Chair with Reinhard Posch from Austria as Vice-Chair.
- IFIP/Sec 96 Samos Island, Greece: "Information systems security: facing the information society of the 21st century"
- IFIP/Sec 97 Copenhagen, Denmark: "IT Security in Research and Business"
- IFIP/Sec 98 Vienna/Budapest, Austria/Hungary with the motto "Global IT security " and as part of the IFIP World Computer Congress
- 1999: No IFIP/Sec Conference took place, but a TC-11 meeting was held in Amsterdam (Netherlands) in conjunction with a TC-11 Working Group conference.
- IFIP/Sec 2000 Beijing, China with the motto "Information Security for global information infrastructures "as part of the IFIP World Computer Congress: Geoff Fairall from Zimbabwe was appointed as new Secretary of TC-11.

- IFIP/Sec 2001 Paris, France: "Trusted information: the new decade challenge". At the TC-11 meeting preceding the conference Leon Strous from the Netherlands was elected as new Chair with Kai Rannenberg from Germany as Vice-Chair. Rossouw von Solms from South Africa was appointed as WG coordinator.
- IFIP/Sec 2002 Cairo, Egypt: "Security in the information society: visions and perspectives"
- IFIP/Sec 2003 Athens, Greece: "Security and privacy in the age of uncertainty"
- IFIP/Sec 2004 Toulouse, France with the motto "Security and protection in information processing systems " as part of the IFIP World Computer Congress
- IFIP/Sec 2005 Tokyo-Chiba, Japan: "Security and privacy in the age of ubiquitous computing". Lech Janczewski from New Zealand, representing SEARCC, was appointed as Secretary.
- IFIP/Sec 2006 Karlstad, Sweden: "Security and privacy in dynamic environments"
- IFIP/Sec 2007 Johannesburg-Sandton, South Africa: "New approaches for security, privacy and trust in complex environments". At the TC-11 meeting preceding the conference Kai Rannenberg from Germany was elected as new Chair with Rossouw von Solms from South Africa as Vice-Chair.
- IFIP/Sec 2008 Milano, Italy as part of the IFIP World Computer Congress: At the TC-11 meeting preceding the conference Yuko Murayama from Japan was appointed as WG Coordinator.
- IFIP/Sec 2009 Pafos, Cyprus: "Emerging Challenges for Security, Privacy and Trust"
- IFIP/Sec 2010 Brisbane, Australia with the motto "Security & Privacy – Silver Linings in the Cloud" as part of the IFIP World Computer Congress: At the TC-11 meeting preceding the conference Yuko Murayama from Japan was appointed as Vice Chair in addition to Rossouw von Solms from South Africa.
- IFIP/Sec 2011 Lucerne, Switzerland with the motto "Future Challenges in Security and Privacy for Academia and Industry"
- IFIP/Sec 2012 scheduled for Greece

TC-11's annual IFIP/Sec conferences are established as an integral and well-reputed part of the international Information Security conference scene. The same holds for many Working Group conferences.

3 Main Development Trends in the Field of TC-11

3.1 The 80es

The early eighties were the years when personal computers started to invade people's lives. One saw an increasing concern about several issues like privacy and witnessed the "birth" of computer viruses. The attention for security started to evolve from the closed defence and mainframe environments to business and small computer environments, from confidentiality towards integrity, from technical security to managerial issues. This was clearly an era where establishing a TC dedicated to security was an obvious thing to do. The founders made it clear by the name and aims and scope of TC-11 that security is not limited to computers but encompasses computers,

applications, data and the organization. That was more or less a visionary view because in those days the term computer security was more common than the term information security.

3.2 The 90es

The increasing trend towards distributed systems, and the associated use of communication networks, as well as the tendency to use such systems and networks for more and more highly sensitive applications like electronic commerce and medical applications, catapulted the absolute importance of the securing and protecting of electronic information during storage, processing and transmission right into the forefront of Information Technology research and implementation.

It became clear that a very large number of such systems would not be acceptable if proper solutions would not exist for the security and protection of such systems. Developments in cryptography showed to be essential to provide non-reputability and proof of origin in electronic messages. Without digital signatures, provided by cryptography, electronic purchasing was deemed to be not possible.

Security in distributed systems became known to be much more difficult and complex than in centralized systems. Authentication and Authorization in distributed systems are of extreme importance, and must be given the necessary attention.

New techniques to implement and to specifically manage Information Security were constantly needed, and with the growing complexity of IT systems, the internal control of the systems became ever more important. The same held for the growing role, importance and commitment of senior management of companies, up to Board level, towards the security of their companies' IT systems. Special efforts were needed to provide skilled people to be able to evaluate, address and manage security risks involved in IT systems, and to ensure that such systems are operated within the necessary secure environment.

With the fact that computers became so much more user friendly than before, and so much more were being used by the public in general, a serious effort showed up as being needed to make these people aware of the importance of Information Security on their systems, and to show them the risks if such security measures were ignored.

In the application field, Information Security became ever more essential for the growing use of systems in medical applications. Standardization efforts and cryptology policies in different countries also required attention. All in all, Information Security had never before been a more important and essential part of IT systems and networks.

These developments were reflected in TC-11's work mainly by the expanding activities in the respective TC-11's Working Groups, but also by public statements from TC-11. One statement concerns IFIP's position on crypto policies and was drafted in the second half of the nineties. It reflected that cryptography was a hot topic from a policy point of view and discussions concentrated on questions such as whether governments should have access to the keys in encryption systems used by companies and individuals. A second statement concerned information security assessment and certification and addressed TC-11's opinion that the information security status of IT systems and the information security management of such systems should be assessed against specified standards related to information security management and that members of IFIP should be instrumental to ensure that such standards, for systems and individuals, be harmonized on an international level.

3.3 The Beginning of the New Millennium

The early beginning of the new Millennium was driven by the Internet and mobile Communication becoming more and more mainstream. “E-words” such as “E-Commerce” and “E-Business” became and more popular. While many of them were just buzzwords, as almost everything from the “old” world became e-d there was little doubt that some of these areas would have significant impact on business and society as a whole. Following this it stepwise became clear that trust and confidence in the security and reliability of all those “e-words” was necessary for them to become the success that everybody was hoping (and waiting) for. So many topics within the scope of TC-11 were influential in that respect, e.g. identification and authentication means (biometrics and smart(er) cards), integrity of messages, secure business transactions and payments.

The events of September 11, 2001 pointed strongly to further aspects of security such as cyber terrorism and (critical) infrastructure protection (CIP). Not only did these issues require new technologies or larger scale use of known technologies (biometrics and smart(er) cards again?) they also shed a different light on privacy issues and human aspects.

To address these issues in an effective way even more cooperation between the different IFIP disciplines was required. Topics of most of the TC's became relevant such as topics like software quality (TC-2), training people (TC-3), safety-critical systems (TC-10), and social aspects and human-computer interaction (TC-9 and TC-13). And although those issues may have seemed to be of a technical nature, one could not hide from the fact that cultural and political aspects also do play a role. IFIP had to consider this when addressing the issues and trying to find a way to deal with them in an as “neutral” as possible fashion. New successful WGs such as WG 11.9 Digital Forensics (established 2004) and the trio of WG 11.6: Identity Management, WG 11.10 Critical Infrastructure Protection, and WG 11.11 Trust Management (all three established 2006) reflected these developments.

A related achievement concerned the objective to promote security and protection as essential elements of information processing systems. TC-11 had been successful in this area, which can be measured directly within the IFIP community by the fact that more and more TC's and working groups were including security in their aims and scopes. This also resulted in an increasing cooperation between TC's and working groups on security topics such as the Communications and Multimedia Security (CMS) conferences of TC-6 and TC-11 and the E-Commerce, E-Government and E-Business (I3E) conferences of TC-6, TC-8 and TC-11, and last but not least the joint WG with TC-9 on legal, privacy and social issues (WG 9.6/11.7 IT Misuse and the Law), a very successful example of an active cooperation.

At the same time some “old” issues did not disappear and one did not succeed in eliminating them. Although their “hot” days were over and they were no longer in the focus of attention (with the exception of an occasional short hype), these activities still had and continued to have a significant impact. Hackers and viruses continued to cost society a lot of money and the security professionals kept trying to find ways to limit the effects as much as possible.

Another important issue was attention for developing countries. While IFIP as a whole supported the work of the Developing Countries Support Committee (DCSC) and the World IT Forum (WITFOR) TC-11 was one of the TCs actively participating in these initiatives by e.g. actively strengthening its activities in developing countries and encouraging participation from the respective member societies to also review and revise traditional (maybe "northern" or "western") views.

Moreover in 2002 TC-11 agreed on another statement which contains a request to all member societies of IFIP to urge their relevant government and education bodies to ensure that proper education and certification requirements are set for those people who intend to become information technology security professionals and including those who audit the security of IT systems.

3.4 Current Challenges

Especially the Internet but also other Information and Communication Technology (ICT) systems such as Mobile Communication systems have moved even further on: From popular and established mainstream technologies to the information and communication backbones for many societies and countries and moreover as the essential infrastructures for global and international cooperation.

The rapid and radical movement towards new and Internet based ICT systems was partially supported by the decline of some established technologies, but also by the changing habits of users. It has raised major questions of trust in to the ICT systems and into information security as such and demonstrated the importance of projection of citizens, consumers and their privacy. TC-11 reflected this development in more and more IFIP/Sec mottos since 2003 and moreover with its first TC name change since its inception: In 2007 the term "Privacy" was added to TC-11's name and subsequently the aims and scope were adapted accordingly. This was preferred to simply establishing a new WG on Privacy as the deep and delicate relations between security and privacy were considered where information security sometimes supports privacy and sometimes endangers it. These delicate relations affect the work of most WGs in IFIP TC-11.

The further miniaturisation and the pervasive use of ICT lead WG 11.2 to changing its name from "Small System Security" to "Pervasive Systems Security" reflecting the fact, that almost every aspect of (human) life is now exposed to ICT. This trend also led to the founding of WG 11.12 "Human Aspects of Information Security and Assurance". At the same time it became clear, that Information Security is also important for researchers and in the information systems field leading to a new WG 8.11 / 11.13 "Information Systems Security Research" together with TC-8.

4 The Kristian Beckman Award

TC-11 established the Kristian Beckman Award in 1992 to commemorate the first chair of the committee, Kristian Beckman from Sweden, who had also been responsible for promoting the founding of TC-11 in 1983. This award is granted not more than annually to a successful nominee and is usually presented at IFIP/Sec. The objective

of the award is to publicly recognise an individual, not a group or organisation, who has significantly contributed to the development of information security, especially from an international perspective. However this particular requirement will not necessarily preclude nominations of those whose main achievements have been made on a national level. Many of the awardees can be considered IFIP (TC-11) Pioneers.

TC-11 was honoured to award the Kristian Beckman Award to:

- Harold Highland (USA) in 1993, presented in Toronto (Canada)
- Per Hoving (Sweden) in 1995, presented in Cape Town (South Africa)
- Sushil Jajodia (USA) in 1996, presented in Samos (Greece)
- Donald Davies (UK) in 1997, presented in Copenhagen (Denmark)
- Richard Sizer (UK) in 1998, presented in Vienna and Budapest (Austria and Hungary)
- Willis W. Ware (USA) in 1999, presented in Amsterdam (Netherlands)
- William Caelli (Australia) in 2002, presented in Cairo (Egypt)
- Roger Needham (UK) in 2003, presented in Athens (Greece)
- Jean-Jacques Quisquater (Belgium) in 2004, presented in Toulouse (France)
- William List (UK) in 2005, presented in Tokyo-Chiba (Japan)
- Butler W. Lampson (USA) in 2006, presented in Karlstad (Sweden)
- Pierangela Samarati (Italy) in 2008, presented in Milano (Italy)
- Klaus Brunnstein (Germany) in 2009, presented in Pafos (Cyprus)
- Sebastiaan von Solms (South Africa) in 2010, presented in Brisbane (Australia)
- Ann Cavoukian (Canada) in 2011, to be presented in Lucerne (Switzerland)

5 The Future Role of TC-11

With the rising importance of ICT systems and society's dependability on these systems, the role of TC-11 and its topics has risen significantly over the last years, and is still rising. TC-11 has taken up this challenge and is active on several fronts through its Working Groups, its special conferences to discuss research developments, and other dissemination services to member societies of IFIP and to the international community in general. However a number of challenges remain and are even growing:

- Still relevant security and privacy issues are only considered relatively late in system development processes – and often still too late.
- Security and privacy are "horizontal" subjects and orthogonal to many topics that are cared for by other IFIP TCs.
- In many cases appropriate decisions with regard to security and privacy can only be taken, if the respective (application) context is considered.

Therefore TC-11 is encouraging the inclusion of security and privacy topics in all areas and actively cooperates with other TCs. This will hopefully contribute to a situation, where relevant security and privacy considerations and measures are embedded as a natural topic in all domains rather than coming in late.

6 Contact Information

TC-11 Homepage: www.ifiptc11.org

TC-11 Management:

TC-11 Chair

Prof. Dr. Kai Rannenberg
T-Mobile Chair of
Mobile Business & Multilateral Security
Goethe University Frankfurt
Postfach 66
Grüneburgplatz 1
60629 Frankfurt /Main, Germany
Tel: +49 69 798 34701
Fax: +49 69 798 35004
www.m-chair.net
kai.rannenberg@m-chair.net

TC-11 Vice-Chair

Prof. Dr. Rossouw von Solms
Nelson Mandela Metropolitan University
Institute for ICT Advancement, School of ICT
Summerstrand (North)
P.O. Box 77000
Port Elizabeth 6031
South Africa
Tel: +27 41 504 3604
Fax: +27 41 504 3313
rossouw.vonsolms@nmmu.ac.za

TC-11 Vice-Chair & WG Coordinator

Prof. Dr. Yuko Murayama
Faculty of Software and Information Science
Iwate Prefectural University
152-52 Sugo, Takizawa,
Takizawa-mura
Iwate, 020-0193
Japan
Tel: +81 19-694-2548
Fax: +81 19-694-2549
murayama@iwate-pu.ac.jp

TC-11 Secretary

Prof. Dr. Lech Janczewski
The University of Auckland
Dept. of ISOM
Private Bag 92019
Owen G Glen Building
12 Grafton Road
Auckland, New Zealand
Tel: +64 9 923 7538
Fax: +64 9 373 7430
lech@auckland.ac.nz

TC-11 Webmaster:

Gökhan Bal
T-Mobile Chair of
Mobile Business & Multilateral Security
Goethe University Frankfurt
Grüneburgplatz 1
60629 Frankfurt /Main, Germany
Tel: +49 69 798 34701, Fax: +49 69 798 35004
www.m-chair.net
contact@ifiptc11.org

Author Index