Igor Kotenko
Victor Skormin (Eds.)

LNCS 7531

Computer Network Security

6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012 St. Petersburg, Russia, October 2012, Proceedings

Lecture Notes in Computer Science 7531

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Igor Kotenko Victor Skormin (Eds.)

Computer Network Security

6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012 St. Petersburg, Russia, October 17-19, 2012 Proceedings

Springer

Volume Editors

Igor Kotenko
St. Petersburg Institute for Informatics and Automation
Russian Academy of Science
39, 14-th Liniya
St. Petersburg, 199178, Russia
E-mail: ivkote@comsec.spb.ru

Victor Skormin
Binghamton University (SUNY)
Binghamton, NY 13902, USA
E-mail: vskormin@binghmaton.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33703-1 e-ISBN 978-3-642-33704-8
DOI 10.1007/978-3-642-33704-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012947925

CR Subject Classification (1998): C.2.0, K.6.5, K.4.4, E.3, D.4.6, C.2.3-4, H.2.7-8, C.5.3, J.1

LNCS Sublibrary: SL 5 – Computer Communication Networks and Telecommunications

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 volume contains papers presented at the 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS 2012) held in St. Petersburg, Russia, during October 17–19, 2012. The conference was organized by the St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS) in cooperation with Binghamton University (SUNY).

The previous international conferences "Mathematical Methods, Models and Architectures for Computer Networks Security" (MMM-ACNS 2001, MMM-ACNS 2003, MMM-ACNS 2005, MMM-ACNS 2007, and MMM-ACNS 2010), organized by SPIIRAS and Binghamton University (SUNY) and supported by the European Office of Aerospace Research and Development USAF, the US Office of Naval Research Global, and the Russian Foundation for Basic Research, were successful. These conferences demonstrated the high interest of the international scientific community in the theoretical and practical aspects of computer network security.

MMM-ACNS 2012 provided the next international forum for sharing original research results among specialists in fundamental and applied problems of computer network security. A total of 44 papers from 12 countries were submitted to MMM-ACNS 2012. Fourteen papers were selected as regular and 8 as short presentations (32% of acceptance for full papers).

Seven technical sessions were organized, namely: applied cryptography and security protocols; access control and information protection; security policies; security event and information management; intrusion prevention, detection, and response; anti-malware techniques; and security modeling and cloud security. The MMM-ACNS 2012 program was enriched by invited papers presented by four distinguished invited speakers: Ben Livshits (Microsoft Research and University of Washington, USA), Fabio Martinelli (Institute of Informatics and Telematics, National Research Council, Italy), Angelos Stavrou (Mason University, USA), and Bhavani Thuraisingham (University of Texas at Dallas, USA).

The success of the conference was assured by the team effort of sponsors, organizers, reviewers, and participants. We would like to acknowledge the contribution of the individual Program Committee members and thank the paper reviewers.

Our sincere gratitude goes to the participants of the conference and all authors of the submitted papers. We are grateful to our sponsors: European Office of Aerospace Research and Development (EOARD) of the US Air Force and the US Office of Naval Research Global (ONRGlobal).

We wish to express our gratitude to Springer's LNCS team managed by Alfred Hofmann for their help and cooperation.

October 2012

Igor Kotenko
Victor Skormin

Organization

General Chairs

Rafael M. Yusupov	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), Russia
Robert L. Herklotz	US Air Force Office of Scientific Research, USA

Program Committee Co-chairs

Igor Kotenko	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), Russia
Victor Skormin	Binghamton University, USA

Program Committee

Fabrizio Baiardi	University of Pisa, Italy
Cataldo Basile	Politecnico di Torino, Italy
Julien Bourgeois	University of Franche-Comte, France
Mariano Ceccato	Fondazione Bruno Kessler, Italy
David Chadwick	University of Kent, UK
Shiu-Kai Chin	Syracuse University, USA
Christian Collberg	University of Arizona, USA
Miguel Pupo Correia	Instituto Superior Técnico, Portugal
Bruno Crispo	University of Trento, Italy
Frédéric Cuppens	Télécom Bretagne, France
Dipankar Dasgupta	University of Memphis, USA
Changyu Dong	University of Strathclyde, UK
Dennis Gamayunov	Moscow State University, Russia
Dieter Gollmann	Technical University of Hamburg-Harburg, Germany
Stefanos Gritzalis	University of the Aegean, Greece
Alexander Grusho	Moscow State University, Russia
Ming-Yuh Huang	Northwest Security Institute, USA
Andrew Hutchison	T-Systems, South Africa
Sushil Jajodia	George Mason University, USA
Angelos Keromytis	Columbia University, USA
Alexey Kirichenko	F-Secure, Finland
Victor Korneev	Federal Enterprise "R&D Institute "Kvant", Russia
Hanno Langweg	Gjøvik University College, Norway
Pavel Laskov	University of Tübingen, Germany

Peeter Laud	Cybernetica AS and University of Tartu, Estonia
Ben Livshits	Microsoft Research and University of Washington, USA
Javier Lopez	University of Malaga, Spain
Antonio Maña	University of Malaga, Spain
Fabio Martinelli	Institute of Informatics and Telematics, National Research Council, Italy
Gregorio Martinez	University of Murcia, Spain
Fabio Massacci	University of Trento, Italy
Catherine Meadows	Naval Research Laboratory, USA
Stig Mjølsnes	Norwegian University of Science and Technology, Norway
Nikolay Moldovyan	SPIIRAS, Russia
Wojciech Molisz	Gdansk University of Technology, Poland
Greg Morrisett	Harvard University, USA
Haris Mouratidis	University of East London, UK
Evgenia Novikova	SPIIRAS, Russia
Vladimir Oleshchuk	University of Agder, Norway
Ludovic Pietre-Cambacedes	EDF, France
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Roland Rieke	Fraunhofer Institute for Secure Information Technology SIT, Germany
Luigi Romano	University of Naples Parthenope, Italy
Andrzej Rucinski	University of New Hampshire, USA
Peter Ryan	University of Luxembourg, Luxembourg
Andrei Sabelfeld	Chalmers University of Technology, Sweden
Ahmad-Reza Sadeghi	TU Darmstadt and Fraunhofer Institute for Secure Information Technology SIT, Germany
Igor Saenko	SPIIRAS, Russia
Francoise Sailhan	CNAM, France
Pierangela Samarati	University of Milan, Italy
Ravi Sandhu	George Mason University and NSD Security, USA
Fred Schneider	Cornell University, USA
Michael Smirnov	Fraunhofer FOKUS, Germany
Angelos Stavrou	Mason University, USA
Nadia Tawbi	Laval University, Canada
Bhavani Thuraisingham	University of Texas at Dallas, USA
Bill Tsoumas	Athens University of Economics and Business, Greece
Shambhu Upadhyaya	University at Buffalo, USA
Paulo Verissimo	University of Lisbon, Portugal
Peter Zegzhda	St. Petersburg Polytechnical University, Russia

Reviewers

Fabrizio Baiardi	University of Pisa, Italy
Cataldo Basile	Politecnico di Torino, Italy
Luciano Bello	Chalmers University of Technology, Sweden
Julien Bourgeois	University of Franche-Comté, France
Mariano Ceccato	Fondazione Bruno Kessler, Italy
David Chadwick	University of Kent, UK
Shiu-Kai Chin	Syracuse University, USA
Christian Collberg	University of Arizona, USA
Miguel Pupo Correia	Instituto Superior Técnico, Portugal
Frédéric Cuppens	Télécom Bretagne, France
Changyu Dong	University of Strathclyde, UK
Dennis Gamayunov	Moscow State University, Russia
Dieter Gollmann	Technical University of Hamburg-Harburg, Germany
Stefanos Gritzalis	University of the Aegean, Greece
Alexander Grusho	Moscow State University, Russia
Ming-Yuh Huang	Northwest Security Institute, USA
Andrew Hutchison	T-Systems, South Africa
Sushil Jajodia	George Mason University, USA
Angelos Keromytis	Columbia University, USA
Alexey Kirichenko	F-Secure, Finland
Victor Korneev	Federal Enterprise "R&D Institute "Kvant", Russia
Hanno Langweg	Gjøvik University College, Norway
Pavel Laskov	University of Tübingen, Germany
Peeter Laud	Cybernetica AS and University of Tartu, Estonia
Ben Livshits	Microsoft Research and University of Washington, USA
Javier Lopez	University of Malaga, Spain
Antonio Maña	University of Malaga, Spain
Fabio Martinelli	Institute of Informatics and Telematics, National Research Council, Italy
Gregorio Martinez	University of Murcia, Spain
Fabio Massacci	University of Trento, Italy
Catherine Meadows	Naval Research Laboratory, USA
Stig Mjølsnes	Norwegian University of Science and Technology, Norway
Nikolay Moldovyan	SPIIRAS, Russia
Wojciech Molisz	Gdansk University of Technology, Poland
Greg Morrisett	Harvard University, USA
Haris Mouratidis	University of East London, UK
Evgenia Novikova	SPIIRAS, Russia
Vladimir Oleshchuk	University of Agder, Norway

Ludovic Pietre-Cambacedes	EDF, France
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Willard Rafnsson	Chalmers University of Technology, Sweden
Roland Rieke	Fraunhofer Institute for Secure Information Technology SIT, Germany
Luigi Romano	University of Naples Parthenope, Italy
Andrzej Rucinski	University of New Hampshire, USA
Peter Ryan	University of Luxembourg, Luxembourg
Andrei Sabelfeld	Chalmers University of Technology, Sweden
Ahmad-Reza Sadeghi	TU Darmstadt and Fraunhofer Institute for Secure Information Technology SIT, Germany
Igor Saenko	SPIIRAS, Russia
Francoise Sailhan	CNAM, France
Pierangela Samarati	University of Milan, Italy
Ravi Sandhu	George Mason University and NSD Security, USA
Fred Schneider	Cornell University, USA
Michael Smirnov	Fraunhofer FOKUS, Germany
Angelos Stavrou	Mason University, USA
Nadia Tawbi	Laval University, Canada
Bhavani Thuraisingham	University of Texas at Dallas, USA
Bill Tsoumas	Athens University of Economics and Business, Greece
Shambhu Upadhyaya	University at Buffalo, USA
Paulo Verissimo	University of Lisbon, Portugal
Peter Zegzhda	St. Petersburg Polytechnical University, Russia

Table of Contents

Invited Papers

Finding Malware on a Web Scale 1
Benjamin Livshits

Exposing Security Risks for Commercial Mobile Devices 3
Zhaohui Wang, Ryan Johnson, Rahul Murmuria, and Angelos Stavrou

From Qualitative to Quantitative Enforcement of Security Policy 22
Fabio Martinelli, Ilaria Matteucci, and Charles Morisset

Design and Implementation of a Cloud-Based Assured Information Sharing System 36
Tyrone Cadenhead, Murat Kantarcioglu, Vaibhav Khadilkar, and Bhavani Thuraisingham

Applied Cryptography and Security Protocols

Optimization of Key Distribution Protocols Based on Extractors for Noisy Channels within Active Adversaries 51
Victor Yakovlev, Valery Korzhik, Mihail Bakaev, and Guillermo Morales-Luna

A Vulnerability in the UMTS and LTE Authentication and Key Agreement Protocols 65
Joe-Kai Tsay and Stig F. Mjølsnes

Blind 384-bit Digital Signature Scheme 77
Alexandr Moldovyan, Nikolay Moldovyan, and Evgenia Novikova

Access Control and Information Protection

RABAC: Role-Centric Attribute-Based Access Control 84
Xin Jin, Ravi Sandhu, and Ram Krishnan

Trust-Aware RBAC 97
Vladimir Oleshchuk

Alternative Mechanisms for Information Security 108
Alexander Grusho, Nick Grusho, and Elena Timonina

Security Policies

Enforcing Information Flow Policies by a Three-Valued Analysis 114
Josée Desharnais, Erwanne P. Kanyabwero, and Nadia Tawbi

Towards the Orchestration of Secured Services under Non-disclosure Policies 130
Tigran Avanesov, Yannick Chevalier, Michaël Rusinowitch, and Mathieu Turuani

An Approach for Network Information Flow Analysis for Systems of Embedded Components 146
Andrey Chechulin, Igor Kotenko, and Vasily Desnitsky

Security Event and Information Management

Individual Countermeasure Selection Based on the Return On Response Investment Index 156
Gustavo Gonzalez Granadillo, Hervé Débar, Grégoire Jacob, Chrystel Gaber, and Mohammed Achemlal

Security and Reliability Requirements for Advanced Security Event Management 171
Roland Rieke, Luigi Coppolino, Andrew Hutchison, Elsa Prieto, and Chrystel Gaber

Model-Based Security Event Management 181
Julian Schütte, Roland Rieke, and Timo Winkelvos

Intrusion Prevention, Detection, and Response

Using Behavioral Modeling and Customized Normalcy Profiles as Protection against Targeted Cyber-Attacks 191
Andrey Dolgikh, Tomas Nykodym, Victor Skormin, and Zachary Birnbaum

Limitation of Honeypot/Honeynet Databases to Enhance Alert Correlation 203
Yosra Ben Mustapha, Hervé Débar, and Grégoire Jacob

Stochastic Model of Interaction between Botnets and Distributed Computer Defense Systems 218
Dmitry P. Zegzhda and Tatiana V. Stepanova

Anti-malware Techniques

Malware Characterization Using Behavioral Components 226
Chaitanya Yavvari, Arnur Tokhtabayev, Huzefa Rangwala, and Angelos Stavrou

MADAM: A Multi-level Anomaly Detector for Android Malware 240
Gianluca Dini, Fabio Martinelli, Andrea Saracino, and Daniele Sgandurra

Using Low-Level Dynamic Attributes for Malware Detection Based on Data Mining Methods 254
Dmitry Komashinskiy and Igor Kotenko

Security Modeling and Cloud Security

Configuration-Based Approach to Embedded Device Security 270
Vasily Desnitsky, Igor Kotenko, and Andrey Chechulin

A Study of Entropy Sources in Cloud Computers: Random Number Generation on Cloud Hosts 286
Brendan Kerrigan and Yu Chen

Security Modeling of Grid Systems Using Petri Nets 299
Peter D. Zegzhda, Dmitry P. Zegzhda, Maxim O. Kalinin, and Artem S. Konoplev

Using Graph Theory for Cloud System Security Modeling 309
Peter D. Zegzhda, Dmitry P. Zegzhda, and Alexey V. Nikolskiy

Author Index 319

Finding Malware on a Web Scale

Benjamin Livshits

Microsoft Research

In recent years, attacks that exploit vulnerabilities in browsers and their associated plugins have increased significantly. These attacks are often written in JavaScript and millions of URLs contain such malicious content.

Over the last several years, we have created a series of techniques designed to detect and prevent malicious software or *malware*. These techniques focus on detecting malware that infects web pages. Much of this research has been done in close collaboration with a major search engine, Bing, which is interested in making sure it does not present malicious results to its users, independently of the user's browser, location, or operating system. As such, detection needs to be as general and wide-reaching as possible. While some of the techniques summarized below can be deployed within a web browser, our primary deployment model involves crawling the web in an effort to find and blacklist malicious pages.

In the rest of this paper, we will summarize three related projects: Nozzle, Zozzle, and Rozzle. Nozzle is a runtime malware detector. Zozzle is a a mostly static malware detector. Finally, Rozzle is a de-cloacking technique that amplifies both.

1 Nozzle: Runtime Heap Spray Detector

Nozzle [3] is a runtime monitoring approach that detects attempts by attackers to spray the heap. Nozzle uses lightweight emulation techniques to detect the presence of objects that contain executable code. To reduce false positives, we developed a notion of global "heap health".

The Nozzle lightweight emulator scans heap allocated object data to identify valid x86 code sequences, disassembling the code and building a control flow graph. Because the attack jump target cannot be precisely controlled, the emulator follows control flow to identify basic blocks that are likely to be reached through jumps from multiple offsets into the object.

We have developed a novel approach to mitigate this problem using global heap health metrics, which effectively distinguishes benign allocation behavior from malicious attacks. Fortunately, an inherent property of heap spraying attacks is the fact such attacks affect the heap globally. Consequently, Nozzle exploits this property to drastically reduce the false positive rate.

2 Zozzle: Mostly Static JavaScript Malware Detector

Zozzle [1] is a low-overhead solution for detecting and preventing JavaScript malware that is fast enough to be deployed in the browser.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 1–2, 2012.

Our approach uses Bayesian classification of hierarchical features of the JavaScript abstract syntax tree to identify syntax elements that are highly predictive of malware.

Our experimental evaluation shows that Zozzle is able to detect JavaScript malware through mostly static code analysis effectively. Zozzle has an extremely low false positive rate of 0.0003, which is less that one in quarter million. Despite this high accuracy, the Zozzle classifier is very fast, with a throughput at over 1 MB of JavaScript code per second.

3 Rozzle: Multi-execution Approach for Revealing Cloaking JavaScript Malware

While static and runtime methods for malware detection been proposed in the literature, both on the client side, for just-in-time in-browser detection, as well as offline, crawler-based malware discovery, these approaches encounter the same fundamental limitation. Web-based malware tends to be environment-specific, targeting a particular browser, often attacking specific versions of installed plugins.

This targeting occurs because the malware exploits vulnerabilities in specific plugins and fail otherwise. As a result, a fundamental limitation for detecting a piece of malware is that malware is triggered infrequently, only showing itself when the right environment is present. In fact, we observe that using current fingerprinting techniques, just about any piece of existing malware may be made virtually undetectable with the current generation of malware scanners.

Rozzle [2] is a JavaScript multi-execution virtual machine, a way to explore multiple execution paths within a single execution so that environment-specific malware will reveal itself. Using large-scale experiments, we show that Rozzle increases the detection rate for offline runtime detection by almost seven times.

In addition, Rozzle triples the effectiveness of online runtime detection. We show that Rozzle incurs virtually no runtime overhead and allows us to replace multiple VMs running different browser configurations with a single Rozzle-enabled browser, reducing the hardware requirements, network bandwidth, and power consumption.

References

Curtsinger, C., Livshits, B., Zorn, B., Seifert, C.: Zozzle: Low-overhead mostly static JavaScript malware detection. In: Proceedings of the Usenix Security Symposium (August 2011)

Kolbitsch, C., Livshits, B., Zorn, B., Seifert, C.: Rozzle: De-cloaking internet malware. In: IEEE Symposium on Security and Privacy (May 2012)

Ratanaworabhan, P., Livshits, B., Zorn, B.: Nozzle: A defense against heap-spraying code injection attacks. In: Proceedings of the Usenix Security Symposium (August 2009)

Exposing Security Risks for Commercial Mobile Devices

Zhaohui Wang, Ryan Johnson, Rahul Murmuria, and Angelos Stavrou

Department of Computer Science
George Mason University, Fairfax VA 22030, USA

Abstract. Recent advances in the hardware capabilities of mobile handheld devices have fostered the development of open source operating systems and a wealth of applications for mobile phones and tablet devices. This new generation of smart devices, including iPhone and Google Android, are powerful enough to accomplish most of the user tasks previously requiring a personal computer. Moreover, mobile devices have access to Personally Identifiable Information (PII) from a full suite of sensors such as GPS, camera, microphone and others.

In this paper, we discuss the security threats that stem from these new smart device capabilities and the online application markets for mobile devices. These threats include malware, data exfiltration, exploitation through USB, and user and data tracking. We present our ongoing research efforts to defend or mitigate the impact of attacks against mobile devices. Our approaches involve analyzing the source code and binaries of mobile applications, kernel-level and data encryption, and controlling the communication mechanisms for synchronizing the user contents with computers and other phones including updates or new version of the operating system or applications over USB. We also explain the emerging challenges in dealing with these security issues when the end-goal is to deploy security-enhanced smart phones into military and tactical scenarios.

1 Introduction

The need for smaller, faster, portable devices and the ever increasing use of technology in our everyday life has driven the hardware manufacturers towards handheld mobile devices that can offer a wide-range of functionality with affordable cost. Newly developed smart gadget devices produced by Apple, Google, Samsung, HTC are powerful enough to accomplish most of the tasks that previously required a personal computer. To make matters worse, unlike most desktop or laptop computers, they are almost always connected to the network. This newly acquired computing power gave a rise to plethora of applications that attempt to leverage the new hardware. These include but are not limited to Internet browsing, email, messaging, social networking, and GPS navigation.

Unfortunately, although powerful and ubiquitous, researchers and practitioners have only recently been looking into the potential threats that stem from

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 3–21, 2012.

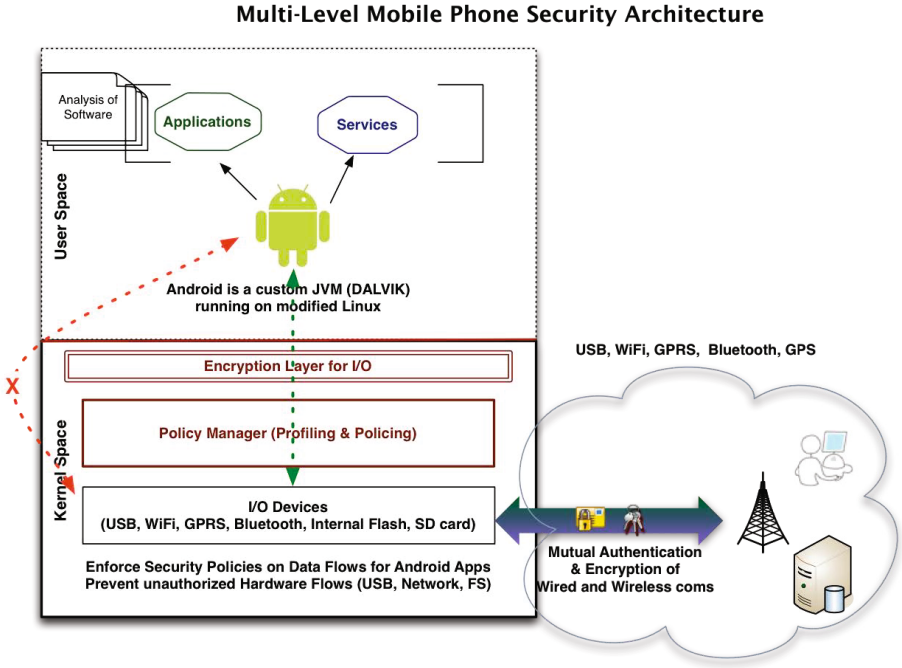

Fig. 1. Overall Security Architecture for Android Devices: the primary design tenet is to prevent Data exfiltration or loss from unauthorized communications and malicious or badly designed mobile applications. A secondary goal was to produce a system that is transparent with small operating footprint in terms of power, CPU, and memory.

device and application attacks on mobile devices. In this paper, we describe the rationale behind some of our efforts [1,3,2] to secure the hardware and software on Android devices used in adversarial environments. Our efforts are data-centric and is multi-pronged as depicted in Figure 1. One of our primary goals is to provide transparent government grade data-at-rest encryption. An Encrypted File System (EncFS) for Android that employees NIST validated crypto algorithms was employed to meet this need.

On the other hand, we wanted to protect information that enters or leaves the mobile device and to prevent unauthorized data leaks. To achieve that, we employ cryptographic communications to all the allowed paths including the USB communications and Internet connections. Finally, to prevent information leakage from untrusted applications, we developed offline security software testing algorithms for Android applications that enable us to weed-out potentially unwanted program functionality that can be construed as malicious depending on the mission requirements.

All the above solutions, and especially encryption, however, come at a potentially significant performance cost depending on the device we apply them on. In general, on mobile devices resources, including battery and processing power are severely constrained so it is important to maintain a small operational footprint.

Throughout this paper, we show that our proposed solutions as depicted in the overall architecture 1 are offering a reliable and secure platform for deployment of missing critical Android applications even when deployed in hostile or any other high risk environments.

2 Background and Related Work

This section provide some background information on the different research solutions that have been proposed over the last few years and illustrates the difficulties to provide an overarching approach to protecting Android mobile devices against a wide-range of attacks.

Mobile OS Attacks and Defenses: The emerging threats brought by smart gadget devices and defense approaches are also well studied by the research community. The presentation "Understanding Android's Security Framework" [4] presents a high-level overview of the mechanisms required to develop secure applications within the Android development framework. The tutorial contains the basics of building an Android application. However, the described interfaces must be carefully secured to defend against general malfeasance. They showed how Android's security model aims to provide mechanisms for requisite protection of applications and critical smart phone functionality and present a number of best practices for secure application development within the environment. However, authors in [5] showed that this is not enough and that new semantically rich and application-centric policies have to be defined and enforced for Android. Moreover, in [6] the authors show how to establish trust and measure the integrity of application on mobile phone systems. TaintDroid [7] addresses the security issues with dynamic information flow and privacy on mobile handheld devices by tracking application behavior to determine when privacy-sensitive information is leaked. This includes location, phone numbers and even SIM card identifiers, and to notify users in realtime. Their findings suggest that Android, and other phone operating systems, need to do more to monitor what third-party applications are doing when running in smart phones. Felt et al. [8] performed testing of Android 2.2.1 in order to identify the Android API calls, intents, and content providers which require a permission.

Battery-Borne Deny-of-Service: Racic and Kim et al. [9,10] studied malware that aims to deplete the power resources on the mobile devices. The provided solutions involve changes in the GSM telephony infrastructure. Their work shows that attacks were mainly carried out through the MMS/SMS interfaces on the device. In addition, in [11] the authors show that applications can simply overuse the WiFi, Bluetooth or display of the device and eventually cause a denial of service attack. VirusMeter [12] models the power consumption and detects the malware based on power abnormality. However, the use of linear regression model with static weights for devices' relative rate of battery consumption is a non-scalable approach [13].

Mobile Malware and Rootkits: Given the popularity of mobile application and their strong coupling relation with PII (Personal Identifiable Information),

the spreading of mobile malware is becoming an alarming threat to military personnel as well as civilians. The evolution of mobile malware created a need to systematically characterize them from various aspects including their installation methods, activation mechanisms as well as the nature of carried malicious payloads given a nearly two years time window [14]. Zhou et al. [15] developed a program to analyze the bytecode of an Android application to create behavioral footprints on Android application and then use heuristics to detect classes of malware.

Cloaker [16] is a non-persistent rootkit that does not alter any part of the host operating system (OS) code or data, thereby achieving immunity to all existing rootkit detection techniques which perform integrity, behavior and signature checks of the host OS. Cloaker leverages the ARM architecture design to remain hidden from current deployed rootkit detection techniques, therefore it is architecture specific but OS independent. Bickford et al. [17] uses three example rootkits to show that smart phones are just as vulnerable to rootkits as desktop operating systems. However, the ubiquity of smart phones and the unique interfaces that they expose, such as voice, GPS and battery, make the social consequences of rootkits particularly devastating.

Code Injection: Buffer overflows also plague mobile devices. The presentation on hacking Windows Mobile [18] at Xcon 2005 talked shell code development advice as well as sample code. Recent emerging threats show that such exploitations are targeting web browsers and other potentially exploitable software like adobe pdf view application in the mobile OSes. Android platform also exposed multiple vulnerabilities for code injection attacks such as CVE-2011-3874 etc. Bojinov et al. proposed a mechanism of executable ASLR that requires no kernel modifications for defending remote code injection attacks for mobile devices [19].

Static Analysis and Execution: There is a plethora of research on static and dynamic analysis of programs with more notable symbolic execution [20]. Most of the analysis programs focus primarily on determining if an Android application requests the correct set of permissions based on the Android API calls that the applications perform [8]. In addition, static analysis programs usually require access to the source code of the Android application. One of these static analysis programs that executes on source code [21] focuses specifically on certain types of behaviors, vulnerabilities, and limited analysis of the permissions. Vidas et al. [22] created a static analysis tool which detects when an Android application requests permissions that it does note need as well as needed but absent permissions from the AndroidManifest.xml file of and Android application. They also developed a plugin for Eclipse which informs developers when they request unneeded permissions based on the application's functionality. They developed a mapping based on the documentation of the Android API. This documentation for the Android API is incomplete, so their mapping of Android API calls is also incomplete. Blasnig et al. [23] use the Android emulator that comes with the Android SDK to perform dynamic analysis on Android application and use a tool to simulate user interaction. The tool also performs some static analysis by disassembling the Android application and identifying certain functionality.

The random-input generation helps to traverse various paths through the code, although using symbolic execution would inform the random-input generator as exactly what inputs would be needed to reach a particular branch of code that has interesting behavior. Moreover, Burguera et al. [24] also use a sandbox to perform dynamic analysis of Android applications and use a behavior-based approach to classify malware by examining the system calls of that the application makes.

3 Motivation

In this section, we discuss the problems and weaknesses we found while researching commodity mobile systems which leads to our proposed solution in next section.

3.1 Open USB Communication

Traditionally, a smart phone device is connected to the host as a peripheral USB device. Being lack of intelligence and computation power, the device is more prone to be taken over by a compromised computer or abused as malware propagation medium. However, the potential attack surface is much wider: the USB creates a bidirectional communication channel, ideally permitting exploits to traverse both directions. Most USB devices are dumb storage medium or only has limited 8bit computation ability such as keyboard. However, new generation phones are equipped more advanced CPUs with complete operating systems which make them as powerful as a traditional desktop system. These recent hardware advancements enables such USB peripherals to perform attacks that are far beyond their ancestors with no chips in terms of computational and software capabilities. Additionally, unlike desktop computers and servers that do not move their physical location, the mobility nature of the smartphones empowers them to potentially communicate to an even larger number of uninfected devices across a wider range of administrative domains. For example, a smart phone left unattended for a few minutes can be completely subverted and become an point of infection to other devices and computers. Lastly, because USB-borne attacks have not been seen before, there are no defenses in place to prevent them from taking place or even detect them.

Currently, USB connections are inherently trusted by the users. When USB protocol was designed decade ago, the physical proximity of the device and the desktop system attributed to such assumed trust based on the fact that, in most cases, the same user owns both systems. However, Wang et. al demonstrated this trust can be easily abused by a malicious adversary [1]. For instance, an unsuspected user connects the smart phone device to the desktop computer to synchronize the two devices including her contact list, media content, calendar and to charge its battery. There are several communication setup steps happening in the systems but all of these are performed completely transparently to the user or with minimal user interaction: the simple press of a mouse click upon

connecting the USB cable. To make matters worse, the usb host(a desktop computer in most of the case) is completely unaware of the type of the device that is connected to the USB port. The usb peripherals can arbitrarily report itself as any usb device given the crafted usb id. This observation can be exploited by a sophisticated adversary who already gained access of the smartphone to launch attacks against the desktop system. Furthermore, there are no mechanisms to authenticate the validity of the device that attempts to communicate with the host in current USB protocol. The lack of authentication allows the connecting peripheral device to disguise and report itself as any type of USB device it want to be, abusing the ubiquitous nature operating system. While the open-medium problem for bluetooth and WiFi has been address in protocol design phrase so that the communication are protected, the USB communication implying a closed-medium do still has the open-medium problem given that the two parties of the communication can not authenticate each other. Our goal is protecting the devices as well as the host from such attacks by applying access control mechanisms on the USB protocol. We refer the USB host as the host system or host side while the USB device as gadget or just device side in the following sections of this paper.

3.2 Lack of Protection for Data at Rest

The recent surge in popularity of smart handheld devices, including smartphones and tablets, has given rise to new challenges in protection of Personal Identifiable Information (PII). Handheld devices are being manufactured all over the world and millions of devices are being sold every month to the consumer market with increasing expectation for growth and device diversity. The price for each unit ranges from free to eight hundred dollars with or without cellular services. In addition, new smartphone devices are constantly released to the market which results the precipitation of the old models within months of their launch. With the rich set of sensors integrated with these devices such as GPS, bluetooth and WiFi, the data collected and generated are extraordinarily sensitive to user's privacy. Indeed, modern mobile devices store PII for applications that span from daily emails to SMS, and from social sharings to location history increasing the concerns of the end user's privacy. Smartphones are therefore data-centric, where the cheap price of the hardware and the significance of the data stored on the device challenge the traditional security provisions. Due to high churn of new devices it is compelling to create new security solutions that are hardware-agnostic. Therefore, there is a clear need and demand for PII data to be protected in the case of loss, theft, or capture of the hardware.

While the application sandboxing protects application-specific data from being accessed by other applications on the phone, sensitive data may be intentionally exfiltrated by malicious code via one of the communication channels such as USB, WiFi, Bluetooth, NFC, cellular network etc. It also can be leaked accidentally due to improper placement, resale or disposal of the device and its storage media (e.g. removable sdcard). Moreover, by simply capturing the smartphones physically, adversaries have access to confidential or even classified

data if the owners are the government officials or military personnels. There is no government standard to regulate and guide the use of smart devices yet. Given the cheap price and rapid evolution of the hardware, the *data* on the devices are more critical and can cause devastating consequences if not well protected. To protect the secrecy of the data through its entire lifetime, we must have robust techniques such as encryption to store and delete data while keeping confidentiality.

We assume that an adversary is already in control of the device or the bare storage media in our threat model, . The memory-borne attacks and defences are not discussed in this paper and addressed by related researches in Section 2 and discussed later. A robust data encryption infrastructure provided by the operating system can help preserve the confidentiality of all data on the smartphone, given that the adversary cannot obtain the cryptographic key. Furthermore, by destroying the cryptographic key on the smartphone we can make the data practically irrecoverable. Our encryption filesystem protects the static data on storage in complimentary to dynamic information flow leaking [7]. Having established a threat model and listed our assumptions, we detail the steps to build encryption filesystem on Android in the following sections.

3.3 Missing Fine-Grain Application Auditing and Regulation

Permission Model on Android platform created debated situation in both industry and academic community. Is such permission model really capable of regulating the applications on mobile operating system and protecting average user's data?There are a couple of studies showing that such permission model, as scatter throughout the whole Android API, failed the aforementioned design goal. In particular, mobile malwares have made their way to bypass such permission model by using other existing applications' capabilities to delegate the malicious behavior. In another work, good app with legit permissions can behave bad. Furthermore, malicious app can also utilizing reflection [25] to evade the permission checking system. Moreover, current permission system is a bipolar system: the user can either grant *all* or deny *all* permissions asked by an application. Such inflexible approach impeded the advanced user to fine-grain auditing and regulating the behavior of the application. We believe that a proper static and dynamic analysis infrastructure will assist both smartphone users and application developers to understand the applications' footprint on filesystems, network and other subsystems. The analysis results can lead to malware discovery and better application design.

4 Proposed Solutions

In previous section, we exposed the missing component in commodity mobile systems. We address those problems by proposing our solutions to tackle them in this section.

4.1 USBSec: Authentication for USB Communication

We outline the design principles we follow and the detailed design of two types of USBSec, a passcode approach and a public key based approach.

Design Principles. Our principle is using easiest engineering, modify minimum set of USB protocol, to achieve reasonable security enhancement including identity authentication, connection authorization. Our design philosophies for both USBSec I and USBSec II are outlined as follows:

- *The authentication is device driver agnostic.* There are a variety of different USB host controllers and USB peripheral controllers in the consumer's market. The design does not depend on any specific device or device driver to accomplish our goal. The authentication logic happens at USB protocol level so that any host controller driver or peripheral controller should have such USB authentication capability when a modified OS kernel with USBSec loads up.
- *No USB hardware modification.* Although the modification is at USB protocol level, the hardware signalling and interrupts remain intact. Only operating system software level changes make the deployment process of USBSec time and cost efficient.
- *Our design is backward compatible.* All existing USB hardware can be used as normal if the peripheral are not listed as USBSec required. The host selectively activates USBSec by configuration listed device vendorID and productsID, when it initiates the USB connection. This is critical to those non-programmable devices which implements USB protocol in the hardware, i.e. USB keyboard and mouse. Our modification to USB enumeration process is compatible with all standard USB devices. In another word, standard USB handshake proceeds if the peripheral device is not listed as authentication required.
- *The authentication is per device.* In the case of a composite USB gadget device, multiple interfaces are available for communication with the host at the same time. The per-device authentication design guarantees that no interface can performance potential malicious action until the authentication of the device is finished.

Implementation. We have implemented a fully working prototype of USBSec I and USBSec II. Our evaluation platform consists of a Dell PowerEdge 1950 server equipped with two Quad-Core Intel Xeon E5430 processors, 16GB RAM as the host; a HTC Nexus One phone and a Motorola Droid phone as the devices. We evaluated USBSec on both devices to show that our design and implementation are not tied to specific hardware controller. The host has the Intel 631xESB/632xESB/3100 chipset as the USB host controller. The devices have the msm_72k OTG controller integrated in QSD8250 SoC with a 1 GHz CPU and ISP1301 USB OTG controller integrated in the OMAP 3430 SoC with a 600 MHz CPU respectively. Both devices run a 2.6.32 based android kernel

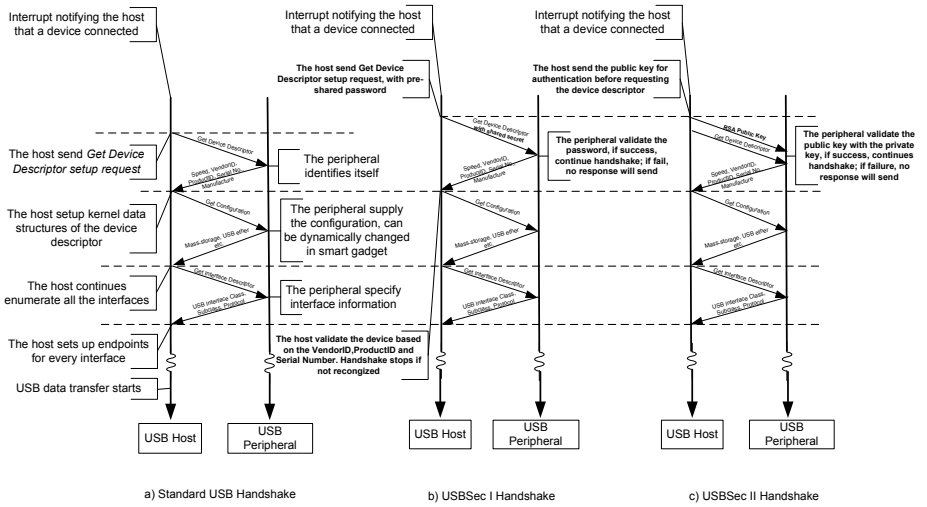

Fig. 2. USB Handshake Diagrams

with respect of different SoC support. The host is running Ubuntu 10.04 with a 2.6.32 kernel. Although we have user-space programs to set the necessary configuration for USBSec, all USBSec processing logic are implemented tightly with the existing USB stack in the Linux kernel. For instance, in USBSec I, the shared passcode is configured via */proc* filesystem by user-space utility, both on the host and the device. Similarly in USBSec II, the keys are generated by *openssl* suite version 0.9.8k program and stored in file system. We wrote the user-space daemon program to load the keys when system boots up and pass it to kernel data structures. Unlike USBSec I, the passcode can be changed at system runtime, the keys in USBSec II only loads at system bootstrapping and only have corresponding kernel memory footprint at runtime. In another word, reconfiguration of the keys requires the system reboot. Specifically, our user-space daemon program will decode the PEM format of the public and private keys to DER(binary) format using *Base64* decoding algorithms and pass it to pre-allocated data structures in kernel. The kernel is responsible to do the Diffie-Hellman key exchange using asymmetric crypto primitives to establish the session keys. However, in mainstream kernel, there is no RSA functionality support yet. We merged the existing RSA kernel implementation [26] with our additional modification to accept the DER binary format keys as input in the gadget kernel to achieve in kernel RSA cipher support. The key size is selected as 1024 bits to tradeoff moderate cryptographic security with performance.

We performed experiments in order to quantify the performances and compare it with traditional USB connection scheme. We measure the connection setup time of USBSec I and USBSec II against standard USB. The start of connection time is define as the time that the host receives the interrupt from USB receptacle notifying the kernel that a device is being connected. The end of connection

time is defined as the time that the host finishes learning the device descriptor and starts requesting the configuration information. Both of them are indicated by a kernel log entry. This time interval includes all our authentication and validation extensions to USB stack. Figure 3 shows our experiments result of USBSec extensions to USB protocol on the two different devices. The analysis result of the data can be highlighted as follows:

- Our USBSec extension only incurs very small amount overhead in connection time and do not affect users' experience. Both of the devices complete the handshake within 2 seconds.
- Handshake takes different amount of time on difference devices, due to different USB controller model. The results reveal that Nexus One takes less time accomplishing the USB handshake than Droid, even in standard USB protocol. USB peripheral controller hardware and the device driver cause the difference.
- Major CPU frequency also plays a key role in the Diffie-Hellman key exchange to establish asymmetric keys. The worst case scenarios reveal that the 1 GHz Nexus One performs faster than the 600 MHz Droid for the same amount of iterations, 256 times in our case.

USB Connection Time(ms)	Standard	USBSec I		USBSec II	
		Best Case	Worst Case	Best Case	Worst Case
Nexus One	254.6ms	343.1	346.7ms	471.1ms	578.9ms
Moto Droid	322.0ms	1373.8ms	1378.2	1744.0ms	1908.6ms

Fig. 3. USBSec Connection Time

Discussion. Linux kernel uses a single bit to disable/enable a USB device on the USB bus [27], providing a basic authorization mechanism. In depth, the kernel will set the device descriptor to "n/a (unauthorized)" and disable it by removing the device configuration information. However, the this scheme has fundamental flaws. First of all, all wired USB devices are authorized by default. In addition , such authorization happens only after the device being connected to the host and the host already enumerated all the interfaces in the devices on the USB bus. It requires a human-interactive operation to explicitly de-authorize the device afterwards. The malicious program running on the device has more than enough time to compromise the host during this gap. Furthermore, this scheme can only authorize the device on the host. From the device's point of view, there is no mechanism to authenticate the identity of the host. For example, a smartphone containing sensitive information can not defend itself from being connected to an unidentified host. Moreover, experiments show that when the user disconnect a deauthorized USB device, the kernel panics at usb_disable_endpoint function and the system become unusable. Further kernel code investigation reveals that even when the same device being connected to the host again, it will be authorized by default.

SELinux applies security policies labeling to files, and AppArmor applies the policies to path names. None of them take considerations for devices inside the kernel.

USB 3.0 is planned to allow for device-initiated communications towards the host, which will make the things more complicated for implementing the authentication scheme in the USB stack. However, we believe with moderate engineering effort, the device-initiated communication can also be authenticated by our approach.

Implementation is crucial to the security strength in any crypto system. It has been conclusively shown that textbook RSA is insecure [28,29]. Secure RSA requires that padding scheme must be used before encryption and signing. USBSec II's in kernel RSA implements padding functionality to the basic RSA operations to encrypt(), decrypt(), sign() and verify() methods.

For passive devices that only have USB hardware implementation but no CPU(e.g. storage device, keyboard), full mutual authentication can not be accomplished due to the limitation of computation capability at the device. Nevertheless, we can authenticate the device and logical driver information by the serial number after the first packets exchange, and prompt the user at the host to allow or deny the connection. Such allowance or rejection can be temporary or permanent. In most of the cases, it's difficult to spoof the serial number information in such passive devices. End user's knowledge and approval help secure the connectivity.

Limitation. Like any password based approach, USBSec I is facing brute force attacks. The adversary can exhaust the password space and defeat USBSec I if gained control of the kernel of either side of the USB communication. The second limitation is USBSec I authentication use serial number information specifically tied to the hardware as the gadget side identity. Any hardware or new inventory change will need corresponding updates to the authorized devices whitelist on the host side.

Bear in mind that we are protecting unauthorized access, USBSec is defeated if the host or the device is already being compromised and spoofs the identity. Because at that time, the trusted chain is broken and authentication is useless.

4.2 EncFS for Android

EncFS is selected as the basis for our encryption filesystem.

EncFS is composed of three major components : kernel FUSE library support, user space *libfuse*, and EncFS binaries. In addition, to make an encryption file-system work on Android, a modified bootstrapping and user login was also integrated into the Android operating system.

EncFS uses standard OpenSSL cryptographic libraries in userspace as cryptographic primitives. This gives us various advantages over using a kernel-based cryptographic library. Some of the features of our solution verses other in-kernel encryption approaches [30,31] can be summarized as follows:

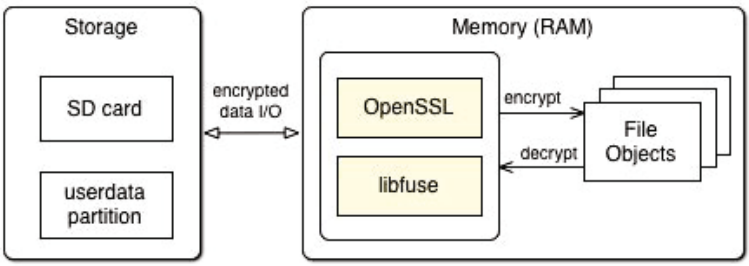

Fig. 4. Abstraction of Encryption Filesystem on Android

- By using EncFS our system is backward and forward compatible with existing and future Android versions. Since *libfuse* and *libc* are stable across different versions of Android and multiple hardware vendors, only minimal engineering efforts if any are needed to make EncFS work on other variations of Android-based smart devices.
- EncFS leverages OpenSSL FIPS suite as the crypto service engine. The OpenSSL libraries, namely libcrypto and libssl, implement cryptographic algorithms that are validated with government FIPS 140-2 Level:1 standard [32].
- In addition, our approach supports different underlying file-systems transparently, including *yaffs2*, *ext4* and *vfat*.

To build EncFS for Android, we created a package with the components described below. It is required the phone has root access at installation time in order to accommodate the kernel with FUSE support, system binaries and java framework patches for integrated login. Once installed, EncFS does not need processes or applications to run as root, in order to encrypt the data. The applications work transparently without knowing underlying changes.

Kernel FUSE Support: FUSE module provides a bridge to the actual kernel interfaces in general. However, the Android Linux kernel does not support FUSE file-systems in early versions. Such minimal kernel configuration reduces fileystem and memory footprints on mobile devices and also eliminate redundant functionalities that are not required by Android. For instance, most Android devices, including the Nexus S which we use, do not come with the FUSE modules enabled in the kernel in off-the-shelf state. We obtain the kernel source code from Google's Android Open Source Project (AOSP) and enabled the kernel FUSE modules necessary for *libfuse* to run. We then flash our device with this customized kernel.

Libfuse: As the required supportive library for all FUSE-based file-systems, *libfuse* is not officially included or supported in the Android system. Moreover, the Bionic C library in Android is a trimmed version of C libarary and missing glue layer code for interfacing VFS (Virtual FileSystem in Linux) and FUSE.

We patched the Bionic C library with missing header files (*statvfs*) and corresponding data structures that are required for *libfuse* version 2.8.5.

EncFS: By building the EncFS executables for the ARM architecture, we created the binaries that would enable us config and manage the EncFS filesystem. In addition to *libfuse*, EncFS also depends on the boost library which is a widely adopted C++ library[33], *librlog* for logging[34] and *libcrypto/libssl* for cryptographic primitives. We patched boost library version 1.45 which is the current-to-date version as of this development and built it against Android Bionic C library. The *librlog* is versioned at 1.4.

EncFS supports two block cipher algorithms: AES and Blowfish. AES runs as a 16 byte block cipher while Blowfish runs as a 8 byte block cipher. Both algorithms support key lengths of 128 to 256 bits and block sizes of 64 to 4096 bytes. Since AES is selected as standard block cipher by US government, our experiments focus on AES only.

Depending on whether we built them as static or shared libraries, we push the binaries onto the system binaries locations on the phone. Figure 5 illustrate the overall layout of EncFS.

User Interface: Normally, the Android framework loads the user interface by unpacking the applications and other files from */system* and */data* partitions. The */data* partition contains all the user-installed applications and all the user-specific data. In our system configuration for EncFS, this */data* partition keeps only a skeleton of the minimal folders to make system bootstrapping. We store the encrypted data in a separate directory and mount it over */data* mountpoint when the user supplies the password.

In addition, we modified the Android Launcher application to accept this password, which is the key for the encrypted version of the */data* partition. If the password provided by the user is valid, EncFS mounts the encrypted data partition on */data* mountpoint. If such mount is performed successfully, the Launcher will call a dedicated native program installed by us to *soft* reinitialize the Android Dalvik environment and the user is presented with his encrypted userdata partition decrypted and loaded into the memory transparently.

To avoid brute-force attacks against the password, the user has a limited number of login attempts. If the failure attempts accumulates to a predefined threshold value (10 in our case), the Launcher program will erase all the encrypted data using secure deleting mechanisms. Although we implemented a program to perform multi-pass secure wipe of the partition, destroying the key alone is adequate as we will be left with a partition full of encrypted data which cannot be decrypted.

Implementation and Performance. Please refer to our full paper [2] for performance details and optimizations.

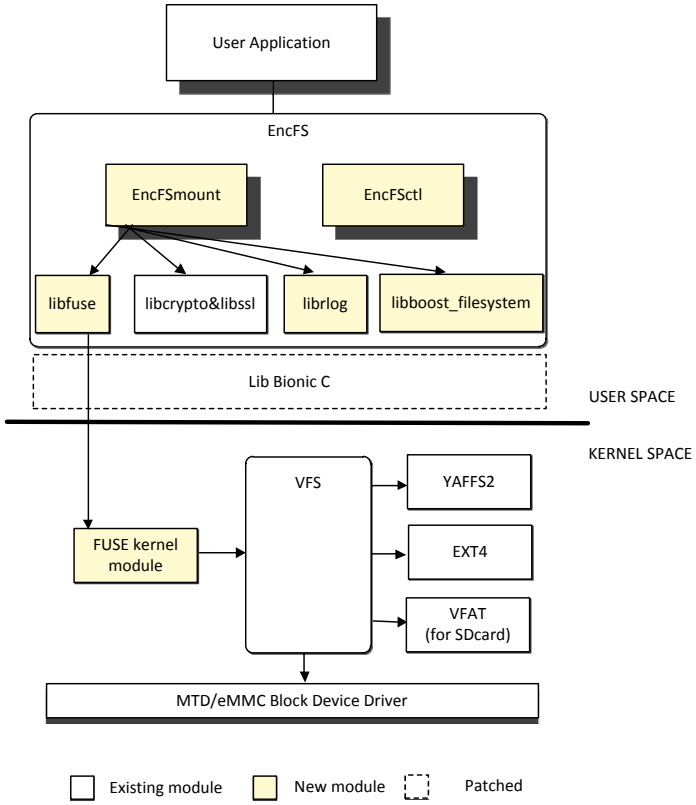

Fig. 5. The operational layout of the Encrypted File System (EncFS)

4.3 Application Analysis

Static analysis serves as a useful method to examine the possible behavior that a program can exhibit. However, static analysis is constrained to certain functionality due to its inherent limitations of not actually executing the code [35]. Static analysis, however, is susceptible to false positives, false negatives, and obfuscation [36,37]. The precision of the analysis increases when the analysis program better understands the semantics of the code and is able to observe the state of the program. When using dynamic analysis, test inputs need to be randomly generated, come from a pre-generated set, or be input by an active entity. Dynamic analysis may or may not get complete coverage of the code, but all the instructions executed will be reachable and the program's true behavior can be observed.

To get as close as possible to complete coverage of the code, the analysis environment has to be able to force the control flow of the program into all potential paths. For instance, as each conditional statement is encountered, either the values of the variables would need to be changed at runtime to obtain the desired

outcome, determined a priori by symbolic analysis, or be forced by controlling the jump to a particular branch independent of the outcome of the boolean condition being evaluated. This type of execution approach [38,39] stresses the program by entering as many branches as possible to make the program exhibit different types of behavior.

The impetus behind this approach is to maximize the coverage of code, as opposed to examining the behavior of the program exhibited by a more limited number of execution traces. This is important because malware can contain very specific conditions that must be met in order for it to display malicious behavior [40]. In certain instances, the behavior is triggered by certain events such as specific times, dates, host names, local IP addresses, the presence of a file, and other factors. In addition, a program may restrain its malicious functionality when it determines that it is being debugged, running in an emulator, or some other type of controlled execution environment [41].

We have developed a set of tools, that runs on a computer and performs concrete execution of an Android application while abstracting certain details from the execution of the application. This abstraction allows the program to automate the analysis of as many paths as possible through the application without requiring any user input. Due to the abstraction, automation is achieved, but the precision of the analysis is reduced. The abstraction is necessary due to not running the application on an Android-enabled phone and the absence of the Android API in disassembled applications. The program only requires an Android Package file (APK) which is the compressed format used to encapsulate an entire Android application into a single file.

Our Android program analysis framework is able to disassemble Android application package files to obtain Dalvik bytecode. Execution of the Dalvik bytecode is possible because we created a Java implementation for each Dalvik bytecode instruction. After using apktool, the disassembled application will be missing the code for the Android API since this code is resident on all Android-enabled phones. Therefore, calls to the Android API are not actually executed unless they are specifically handled by our program which only occurs for a very small set of simple API calls. These were manually coded into our program after examining the specification and exhibited functionality for each handled API call. In addition, we also leverage the Java API since the program runs inside the JVM by creating a wrapper around certain Java API calls.

The values of primitive data types and objects used as parameters to Android API calls to be examined and extracted. These parameters depend on the specific execution path taken through the program. The values of the primitive data types within an object that are used as a parameter to an API call can also be examined. This enables a more through understanding of the program's behavior, and allows hard-coded values in the application to be operated on and extracted. The precision of the analysis is limited due to the lack of user interaction, user input, and Android API which will prevent certain values and objects from being available.

The program allows all Dalvik instructions, method calls, and API calls to be hooked for analysis and monitoring. Currently, the program monitors and records very specific behavior (e.g., commands issued, execution of binaries, use of Java reflection, loading of libraries, network events, files accessed, exhaustive list of methods called, control flow, and the use of dynamic class-loading), although this can be further expanded to anything of interest. The use of Java reflection is recorded because it can help to obfuscate the actual method being called unless the parameters are examined to see exactly which method of what class is being called. Reflection also obviates the use of visibility modifiers in Java [25].

Limitations. Our approach can be computationally expensive depending on the structure and size of the program being analyzed. Each if conditional statement that occurs outside of a loop exponentially raises the number of iterations that must be executed to cover all possible paths within an application. Loops that are nested to a high degree significantly affect the performance of the program due to the large number of iterations through the code, especially when many if conditional statements occur within each loop. An attacker could purposefully plant various computationally expensive activities throughout the program to purposefully slow the analysis of the application. There are approaches to make a trade-off between analysis time and analysis precision. There are the options to limit the number of iterations through a loop, prevent nesting beyond a certain number of loops, limit recursion, or use a timer that sets a maximum time that can elapse to indicate that a path should end.

5 Conclusions

In this paper, we discuss the security threats that arise from new smart device capabilities and brave new world of Android applications deliver over the network to the device. Some of these threats span many known research ares including data exfiltration, exploitation through USB, and user and data tracking.

To address some of these threats, we detail our ongoing research efforts to defend or mitigate the impact of attacks against mobile devices. To that end, we presented data-centric a framework to secure Android devices against attacks that attempt to infiltrate or exfiltrate data, corrupt the operations of the device or perform unwanted functionality. Our approach is multi-pronged and focuses on hardening the device by enabling Kernel-level network and data encryption, and controlling the communication mechanisms for synchronizing the user contents with computers and other phones. We verified our approach by testing a large number of Android applications with our program to exhibit its functionality and viability. The framework allows complete automation of the testing process, so that no user input is required. Given the constant improvements in both hardware and software, we believe that our framework provides a good and adaptable solution for mobile device security.

References

1. Wang, Z., Stavrou, A.: Exploiting smart-phone usb connectivity for fun and profit. In: ACSAC 2010: Annual Computer Security Applications Conference (2010)
2. Wang, Z., Murmuria, R., Stavrou, A.: Implementing & optimizing an encryption file system on android. In: SERE 2012: 6th International Conference on Software Security and Reliability, SERE 2012 (2012)
3. Wang, Z., Johnson, R., Stavrou, A.: Attestation & authentication for usb communications. In: IEEE International Conference on Mobile Data Management, IEEE MDM 2012 (2012)
4. Enck, W., McDaniel, P.: Understanding android's security framework. In: CCS 2008: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 552–561. ACM, New York (2008)
5. Ongtang, M., Mclaughlin, S., Enck, W., Mcdaniel, P.: Semantically rich application-centric security in android. In: ACSAC 2009: Annual Computer Security Applications Conference (2009)
6. Muthukumaran, D., Sawani, A., Schiffman, J., Jung, B.M., Jaeger, T.: Measuring integrity on mobile phone systems. In: SACMAT 2008: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 155–164. ACM, New York (2008)
7. Enck, W., Gilbert, P., gon Chun, B., Jung, L.P.C.J., McDaniel, P., Sheth, A.N.: Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In: OSDI 2010: Proceedings of the 9th Symposium on Operating Systems Design and Implementation, pp. 255–270. ACM, New York (2010)
8. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 627–638. ACM (2011)
9. Radmilo Racic, D.M., Chen, H.: Exploiting mms vulnerabilities to stealthily exhaust mobile phone's battery. In: SecureComm 2006, pp. 1–10 (2006)
10. Kim, H., Smith, J., Shin, K.G.: Detecting energy-greedy anomalies and mobile malware variants. In: MobiSys 2008: Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services, pp. 239–252. ACM, New York (2008)
11. Moyers, B.R., Dunning, J.P., Marchany, R.C., Tront, J.G.: Effects of wi-fi and bluetooth battery exhaustion attacks on mobile devices. In: HICSS 2010: Proceedings of the 2010 43rd Hawaii International Conference on System Sciences, pp. 1–9. IEEE Computer Society, Washington, DC (2010)
12. Liu, L., Yan, G., Zhang, X., Chen, S.: Virusmeter: Preventing your cellphone from spies. In: RAID 2009: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, pp. 244–264. Springer, Heidelberg (2009)
13. Nash, D.C., Martin, T.L., Ha, D.S., Hsiao, M.S.: Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices. In: PERCOMW 2005: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 141–145. IEEE Computer Society, Washington, DC (2005)
14. Zhou, Y., Jiang, X.: Dissecting android malware: Characterization and evolution. In: IEEE Symposium on Security and Privacy, pp. 95–109. IEEE Computer Society (2012)
15. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In: Proceedings of the 19th Annual Network and Distributed System Security Symposium, NDSS (February 2012)

16. David, F.M., Chan, E.M., Carlyle, J.C., Campbell, R.H.: Cloaker: Hardware supported rootkit concealment. In: SP 2008: Proceedings of the 2008 IEEE Symposium on Security and Privacy, pp. 296–310. IEEE Computer Society, Washington, DC (2008)
17. Bickford, J., O'Hare, R., Baliga, A., Ganapathy, V., Iftode, L.: Rootkits on smart phones: attacks, implications and opportunities. In: HotMobile 2010: Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, pp. 49–54. ACM, New York (2010)
18. Phrack: Hacking windows ce, http://www.phrack.org/issues.html?issue=63&id=6
19. Bojinov, H., Boneh, D., Cannings, T.R., Malchev, I.: Address space randomization for mobile devices. In: Fourth ACM Conference on Wireless Network Security (WISEC 2011), pp. 127–138 (2011), http://www.odysci.com/article/1010113016076341
20. King, J.C.: Symbolic execution and program testing. Commun. ACM 19(7), 385–394 (1976)
21. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A Study of Android Application Security. In: Proceedings of the 20th USENIX Security Symposium (August 2011)
22. Vidas, T., Christin, N., Cranor, L.: Curbing Android permission creep. In: Proceedings of the Web 2.0 Security and Privacy 2011 Workshop (W2SP 2011), Oakland, CA (May 2011)
23. Bläsing, T., Batyuk, L., Schmidt, A.D., Camtepe, S., Albayrak, S.: An android application sandbox system for suspicious software detection. In: 2010 5th International Conference on Malicious and Unwanted Software (MALWARE), pp. 55–62 (October 2010)
24. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM 2011, pp. 15–26. ACM, New York (2011)
25. Richmond, M., Noble, J.: Reflections on remote reflection. In: Proceedings of the 24th Australasian Conference on Computer Science, ACSC 2001, pp. 163–170. IEEE Computer Society, Washington, DC (2001)
26. Linux: Rsa kernel patch, http://lwn.net/Articles/228892/
27. Community, L.O.S.: Linux usb authorization, http://lxr.linux.no/linux+v2.6.32.24/Documentation/usb/authorization.txt
28. Boneh, D., Cryptosystem, T.R., Rivest, I.R., Shamir, A., Adleman, L., Rst, W.: Twenty years of attacks on the rsa cryptosystem. Notices of the AMS 46, 203–213 (1999)
29. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
30. Team, G.A.: Android honeycomb encryption, http://source.android.com/tech/encryption/android_crypto_implementation.html
31. Whispercore: Whispercore android device encryption, http://whispersys.com/whispercore.html
32. Project, O.: Openssl fips 140-2 security policy
33. Boost: Boost c++ library, http://www.boost.org/
34. Librlog: Librlog, http://www.arg0.net/rlog
35. Sharif, M.I., Lanzi, A., Giffin, J.T., Lee, W.: Impeding malware analysis using conditional code obfuscation. In: NDSS, The Internet Society (2008)

36. Chess, B., McGraw, G.: Static analysis for security. IEEE Security and Privacy 2(6), 76–79 (2004)
37. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Computer Security Applications Conference, ACSAC, Twenty-Third Annual, 421–430 (December 2007)
38. Wilhelm, J., Chiueh, T.-c.: A Forced Sampled Execution Approach to Kernel Rootkit Identification. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 219–235. Springer, Heidelberg (2007)
39. Lu, S., Zhou, P., Liu, W., Zhou, Y., Torrellas, J.: Pathexpander: Architectural support for increasing the path coverage of dynamic bug detection. In: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO (2006)
40. Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., Yin, H.: Automatically Identifying Trigger-based Behavior in Malware. In: Botnet Analysis. Springer Publications (2007)
41. Moser, A., Kruegel, C., Kirda, E.: Exploring multiple execution paths for malware analysis. In: SP 2007: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pp. 231–245. IEEE Computer Society, Washington, DC (2007)

From Qualitative to Quantitative Enforcement of Security Policy

Fabio Martinelli, Ilaria Matteucci, and Charles Morisset

IIT-CNR, Security Group
Via Giuseppe Moruzzi 1, 56124 Pisa, Italy
firstname.lastname@iit.cnr.it

Abstract. The problem of enforcing a security policy has been particularly well studied over the last decade, following Schneider's seminal work on security automata. We first present in this paper this problem through its qualitative aspect, where one tries to specify and to define a "good" runtime monitor. In particular, we recall that under some conditions, a monitor can be automatically synthesized, using partial model checking. We then introduce some of the quantitative challenges of runtime enforcement, which focus on the problem of defining what does it mean for a monitor to be better than another one, and we sketch several directions that could be explored to tackle this issue.

1 Introduction

In the last years, the security of the systems is one of the main topics to be addressed in computer science. Indeed the amount of information and sensible data that circulate on the Internet and the number of different devices through which it is possible to share information has been growing up till a saturation point. Nowadays, basically each of us is daily equipped with a computer, i.e., our smart-phone. This situation is even more dangerous than previously since the kind of users is wider, and likely less skilled on average. Furthermore, the kind of attacks one can perform is wider since now smart-phones can be used for both work and leisure activities, thus exposing ourselves to an increase of possibilities of private behavior leakage.

This makes more urgent, if possible, the necessity of developing mechanisms for enforcing security policies to be satisfied by our computers and personal devices in order to guarantee the secrecy, the confidentiality and the integrity of our (private) information. For these reasons, a lot of work has been done on the enforcement of security policies on systems and devices, in order to prevent damages to users.

The problem of monitoring a system such that it always satisfies a security policy is a well-studied problem. From a formal perspective, since Schneider's seminal work [32], there has been a lot of effort into characterizing what kind of policy can be monitored and enforced at runtime. This leads to the clear identification of basic concepts, such as security policy, target (or monitored system),

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 22–35, 2012.

execution traces, enforcement mechanisms, etc. This strand of work initially considered only the blocking of the system in case of wrong executions; It has then been refined by considering several approaches for the correction of execution traces by the enforcement mechanisms. The kind of security policies that can be enforced with these models have been extensively studied leading to precise characterization of what can be enforced with a certain class of enforcement mechanisms and under which conditions.

One research strand that was also developed is the adoption of formalisms such as process algebras to better define the interactions between the target and the enforcement mechanism as well as the enforcement capabilities, due to the well studied semantical composition frameworks developed for these formal models. For instance, it is possible to formalize the behavior of a security automata through process algebra operators. By using the techniques and solutions developed for process algebras and also by adopting concepts developed in temporal logic, it is possibly to study the problem of automatically synthesizing enforcement mechanisms for targets (and parts of targets) and policies expressed in temporal logic. The techniques developed using process algebras, partial model checking and satisfiability techniques for logic allow one also to create mechanisms that enforce global security policies on a system, by only considering the target as a subcomponent of this system, thus enabling the possibility to project global security policies onto local ones.

The previous research work seems to be mainly focused on what we call *qualitative* enforcement, i.e., finding a "good" enforcement mechanism for a security policy, if possible. We can also consider the more challenging problem, that we are going to name here *quantitative* enforcement, that amounts to find the "best" enforcement mechanism for a security policy and a target, in accordance to some quantitative criteria suc as cost or precision of enforcement.

On the one hand quantitative enforcement could be seen as an optimization problem to find the best among all good enforcement mechanisms. On the other hand, when a good enforcement mechanism does not exist, for all possible target behaviors, maybe it could be considered as a way to trade-off between security and usability in order to find the best possible enforcement mechanism.

We give here some ideas on how "quantities" can be used in the ambit of enforcement of security policies:

- cost of enforcement (i.e. not all security mechanisms/decisions come for free);
- uncertainty (i.e. systems evolving in a real setting are prone to uncertain parameters, that must be taken into account);
- optimization (i.e. it is not sufficient to define a "good" enforcement mechanisms, we might wish to find the "best" one).

We believe the problem of quantitative enforcement is a complex one, that is only being recently considered, and we do not aim in this paper at providing a complete solution, but rather identifying some of the key aspects that need to be considered and fostering this research strand. As a matter of fact, this kind of problems is particularly relevant since information systems are pervasive, used in many circumstances and under several, often conflicting operation

criteria, of which security is a major one, although not the unique. Thus, we need to embed decision making processes in the loop for security management. Understanding and measuring how the system can cope with "wrong"/"good" decisions is therefore crucial.

The rest of this paper is organized as follows: in Section 2, we present the problem of runtime enforcement, and we show how to define a controller from a security automaton. In Section 3, we show that under some conditions, such a controller can be synthesized automatically. In Section 4, we describe several quantitative dimensions of the runtime enforcement problem. In Section 5, we present some related work, after which we conclude and describe some future directions.

2 Enforcing Security Policies

In the following we first recall some notions about *enforcement mechanisms* and *security automata*. Then we present four process algebra *controller operators* whose behavior mimics the enforcement strategy of security automata. It is worth noticing that, thanks to the power and the flexibility of process algebra, it is possible to define several controller operators able to enforce security policies in different way (see e.g. [24,23]). Furthermore, the usage of process algebra allows us to exploit well-known mechanisms as *partial model checking* and *satisfiability* in order to automatically obtain such controller programs.

2.1 Enforcement Mechanisms and Security Automata

According to [32], an *enforcement mechanism* is a mechanism that works by monitoring a *target system*, that is the system we want to check, and terminating any execution that is about to violate the security policy being enforced. The class EM (Execution Monitoring) includes *security kernels*, *reference monitors*, and other operating systems and hardware-based enforcement mechanisms.

Starting from the generic definition of enforcement mechanisms [32], a *security automaton* is a triple $(\mathcal{Q}, q_0, \delta)$, where $\mathcal{Q}$ is a set of states, q_0 is the initial one and $\delta : Act \times \mathcal{Q} \to \mathcal{Q}$, where Act is a set of actions, is the transition function. A security automaton processes a sequence $a_1 a_2 \ldots$ of actions. At each step only one action is considered and for each action we calculate the *global state* Q' that is the set of possible states for the current action. In other words, if the automaton is checking the action a_i, then $Q' = \bigcup_{q \in Q'} \delta(a_i, q)$. If the automaton can make a transition on a given action, i.e., Q' is not empty, then the target is allowed to perform that step. The state of the automaton changes according to the transition rules. Otherwise, the target execution is terminated. Thus, at every step, it verifies if the action is in the set of the possible actions or not.

This basic definition is used and extended in [6,7], where several possible behaviors of security automata are defined. To compare the power of different enforcement mechanisms, the following two abstract principles are given [7]:

- **Soundness:** An enforcement mechanism must ensure that all observable outputs always obey the policy in question;

– **Transparency:** An enforcement mechanism must preserve the semantics of executions that already obey the policy in question.

2.2 From Security Automata to Process Algebra Controller Operators

Referring to [6,7], in this section we recall the semantics of security automata *truncation, suppression, insertion, edit* and we describe some process algebra operators that model their behavior.

The execution of each different kind of security automaton $\mathbf{K}$, with $\mathbf{K} \in \{T, S, I, E\}$, is specified by a labelled operational semantics.

The automata behaviors differ from one another due their transition functions δ, and these differences account for the variations in their expressive power. The exact specification of δ is part of the definition of each automaton. Roughly speaking, the suppression truncation automaton is able to halt the execution of the target whenever it tries to perform an action that is going to violate the policy. In addition to the truncation automaton functionalities, the suppression automaton is able to hide some actions, the insertion is able to add some actions before a possible malicious action that can be recovered later on in the execution trace, and, the edit automaton groups all these functionalities.

In [26], we have proposed four controller operators by showing their behavior through semantics rules. Each operator mimics one of the security automata. For the sake of space, we only recall here the result for the truncation automata and for the edit automata.

Truncation automaton. The operational semantics of a truncation automaton is given by:

if $\sigma = a;\sigma'$ and $\delta(a, q) = q'$

$$(\sigma, q) \xrightarrow{a}_T (\sigma', q') \qquad \text{(T-Step)}$$

otherwise

$$(\sigma, q) \xrightarrow{\tau}_T (\cdot, q) \qquad \text{(T-Stop)}$$

We write E for the controller program and F for the target. We work, without loss of generality, under the additional assumption that E and F never perform the internal action τ since truncation automata do not consider internal action. We define the controller operators $\triangleright_T$ as follows:

$$\frac{E \xrightarrow{a} E' \;\; F \xrightarrow{a} F'}{E \triangleright_T F \xrightarrow{a} E' \triangleright_T F'}$$

This operator models the truncation automaton that is similar to Schneider's automaton (when considering only deterministic automata, e.g. see [6,7]). Its semantics rule states that if E and F perform the same action, then such action is allowed.

Hence, given a truncation automaton $(\mathcal{Q}, q_0, \delta)$ and the current state q, we define the controller E^q as follows:

$$E^q = \sum_{a \in Act} \begin{cases} a.E^{q'} & \text{if } \delta(a,q) = q' \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Each sequence of actions that is an output of a truncation automaton $(\mathcal{Q}, q_0, \delta)$ is also derivable from $E^q \triangleright_T F$ and vice-versa.

Edit automaton. An edit automaton is defined by a 5-tuple $(\mathcal{Q}, q_0, \delta, \gamma, \omega)$, where $\gamma : Act \times \mathcal{Q} \to Act \times \mathcal{Q}$ specifies the insertion of a finite sequence of actions into the program's actions sequence and $\omega : Act \times \mathcal{Q} \to \{-, +\}$ indicates whether or not the action in question should be suppressed (-) or emitted (+). ω and δ have the same domain, while the domain of γ is disjoint from the domain of δ, in order to have a deterministic automaton. Its operational semantics is given by:

if $\sigma = a; \sigma'$ and $\delta(a, q) = q'$ and $\omega(a, q) = +$

$$(\sigma, q) \xrightarrow{a}_E (\sigma', q') \qquad \text{(E-StepA)}$$

if $\sigma = a; \sigma'$ and $\delta(a, q) = q'$ and $\omega(a, q) = -$

$$(\sigma, q) \xrightarrow{\tau}_E (\sigma', q') \qquad \text{(E-StepS)}$$

if $\sigma = a; \sigma'$ and $\gamma(a, q) = (b, q')$

$$(\sigma, q) \xrightarrow{b}_E (\sigma, q') \qquad \text{(E-Ins)}$$

otherwise

$$(\sigma, q) \xrightarrow{\tau}_E (\cdot, q) \qquad \text{(E-Stop)}$$

In order to both insert and suppress actions, we define the controller operator $\triangleright_E$ as the union of the rules of the $\triangleright_S$ and $\triangleright_I$, defined as follows.

$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E \triangleright_E F \xrightarrow{a} E' \triangleright_E F'} \qquad \frac{E \xrightarrow{-a} E' \quad F \xrightarrow{a} F'}{E \triangleright_E F \xrightarrow{\tau} E' \triangleright_E F'}$$

$$\frac{E \overset{a}{\not\to} E' \quad E \xrightarrow{+a.b} E' \quad F \xrightarrow{a} F'}{E \triangleright_E F \xrightarrow{b} E' \triangleright_E F}$$

This operator combines the power of all the different automata, i.e., truncation, suppression, and insertion automata. Hence, given an edit automaton $(\mathcal{Q}, q_0, \delta, \gamma, \omega)$ and the current state q, we define the controller $E^{q,\gamma,\omega}$ as follows:

$$E^{q,\gamma,\omega} = \sum_{a \in Act} \begin{cases} a.E^{q',\gamma,\omega} & \text{if } \delta(a,q) = q' \text{ and } \omega(a,q) = + \\ -a.E^{q',\gamma,\omega} & \text{if } \delta(a,q) = q' \text{ and } \omega(a,q) = - \\ +a.b.E^{q',\gamma,\omega} & \text{if } \gamma(a,q) = (b,q') \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Each sequence of actions that is an output of an edit automaton $(\mathcal{Q}, q_0, \delta, \gamma, \omega)$ is also derivable from $E^{q,\gamma,\omega} \triangleright_E F$ and vice-versa.

The usage of process algebra controller operators for enforcing security policies provides several advantages in terms of compositional reasoning. Indeed, by using this technique we are able to limit the control to possibly un-trusted components of a given system. Other approaches deal with the problem of monitoring the possible untrusted component to enjoy a given property, by treating it as the whole system of interest. However, often the system does not need to be entirely checked (or it is simply not convenient to check it as a whole). Some components could be trusted and one would like to have a method to control only un-trusted ones (e.g. downloaded applets). Similarly, it could not be possible to build a monitor for a whole distributed architecture, while it could be possible to have it for some of its components. Furthermore, our approach can exploit well-know results enabling us to automatically synthesize a *controller program* for each controller operator (Section 3).

3 Synthesis of Controller Operator

The problem of *synthesis*, first addressed by Merlin and Bochman in [28], occurs when one deals with a system in which there are some unspecified components, e.g., a not completely implemented software. When considering a partially specified system, one may wonder if there exists an implementation that can be plugged into the system, replacing the unspecified one, by satisfying some properties of the whole system. Hence, given a system S, the problem can be formulated as:

$$\exists E \quad S \| E \models \phi$$

where ϕ is a logic formula representing the property to be satisfied.

The problem of the *synthesis of secure systems* is slightly different. Let us consider a system that we want to secure. We can study it as a partially specified system, hereafter referred as *open system* [22]. The unspecified part is a component whose behavior is not known a priori, and we want the system to be secure, whatever the behavior of the unspecified components is. With S the system, F the unspecified component, $S \| F$ the partially specified system, we require that:

$$\forall F \quad S \| F \models \phi$$

where, again, ϕ is a logic formula representing the property.

Since it is not always possible to check all possible behavior of the component F, we develop mechanisms that guarantee the system to work properly by forcing the desired behavior of the unspecified component in such a way that the system satisfies the required formula. Hence, we wonder if there exists an implementation that, by monitoring the behavior of the unspecified component F, guarantees the system to satisfy the required security property:

$$\exists E \quad \forall F \quad S \| (E \triangleright F) \models \phi$$

where $\triangleright$ is a symbol denoting the fact that E monitors the behavior of F.

In particular, we want to solve the synthesis problem for each controller operators $\triangleright_{\mathbf{K}}$ defined above. Hence, we want to solve the following problems

$$\exists E \, \forall F \quad (S \| E \triangleright_{\mathbf{K}} F) \models \phi \tag{1}$$

for each $\mathbf{K} \in \{T, S, I, E\}$.

First of all we apply the *partial model checking* function [1] to evaluate the formula ϕ by the behavior of S. In this way we obtain a new formula $\phi' = \phi_{//_S}$ and we only have to monitor the target F. The formula ϕ' represents the necessary and sufficient conditions that $E \triangleright_{\mathbf{K}} F$ has to satisfy in order to guarantee the security of the system. Indeed, the problem we have to solve is reduced to the following one:

$$\exists E \, \forall F \, (E \triangleright_{\mathbf{K}} F) \models \phi' \tag{2}$$

It is worth noticing that we are going to enforce safety properties. Hence, let us consider a subclass of equational μ-calculus formulas that we call Fr_μ, which consists of equational μ-calculus formulas without the diamond modality. This set of formulas is closed under the partial model checking function and, according to [11], if two processes have a *similar* behavior, they satisfy the same formulas belonging to Fr_μ. According to the semantics of $\triangleright_{\mathbf{K}}$ operators, for every $\mathbf{K} \in \{T, S, I, E\}$, $E \triangleright_{\mathbf{K}} F$ is simulated by E modulo a relabelling function depending on $\mathbf{K}$. Hence whatever controller operators $\triangleright_{\mathbf{K}}$ is chosen to enforce a given safety property, it is possible to find a solution for the problem in Formula 2 by solving the following satisfiability problem:

$$\exists E \quad E \models \phi'_{\mathbf{K}} \tag{3}$$

where $\phi'_{\mathbf{K}}$ is the formula ϕ' modulo the relabelling function of the controller operator $\triangleright_{\mathbf{K}}$. According to satisfiability results of μ-calculus formulas [33], it is possible to find a controller program E that models the formula $\phi'_{\mathbf{K}}$. Furthermore, using a finitary axiom system, proposed by Walukiewicz in [34], it is possible to automatically synthesize E according to the Walukiewicz satisfiability procedure.

4 Quantitative Reasoning

In the previous section, we have presented a qualitative approach to the problem of runtime enforcement, that is, an approach to address the question of whether, given a property, it is possible to define a "good" controller, and if so, how to automatically synthesize it.

In this section, we focus on the *quantitative* aspects of runtime enforcement, that is, we try to define what does it mean for a controller to be "better" than another one, and if there a "best" one. There are several dimensions associated with the notion of quantitative enforcement, and we present in the rest of this section a non-exhaustive list:

– Calculating how much can a monitor be non-secure and/or non transparent (Section 4.2);

- Taking into account how likely the target will behave in the future (Section 4.3);
- Calculating how much a particular enforcement strategy can cost (Section 4.4);
- Calculating how much the system can gain or lose from executing a trace (Section 4.5).

When dealing with quantitative aspects, it is important to distinguish between the decision process and the actual implementation of the security controller. In other words, in an analogous way as it is done in XACML [30], we wish to distinguish between the decision point and the enforcement point. Hence, we introduce in Section 4.1 the concept of *selector*, which is a function indicating at each step what decision should be made, and we show how to build a security controller from such a function.

4.1 Trace Selector

As we have already said in Section 2.2, a security automaton, such as an edit-automaton, consists of a security state and several transition functions, each function being responsible for a particular aspect of the trace manipulation (inserting, accepting, suppressing, stopping). Hence, an edit automaton models at the same time how should each action input by the target be treated, and how the global controller evolves.

Hence, it is worth observing that a security controller needs to make a *decision*, and that some usual techniques in decision theory can be used in order to help this process. In order to embody the decision making, we first introduce a set $\mathcal{D}$ of *atomic decisions*, such that each decision represents an action the security controller can do at each step. For instance, we could model the truncation automaton by defining $\mathcal{D} = \{acc, stop\}$, meaning that at each step, the security controller can only either accept the input or stop.

At first glance, we could define a selector as a function taking an action and returning a decision. Many access control systems can be defined in this way, such as the Role-Based Access Control (RBAC) [16] model, where an action is simply a request for access from a user over a resource, and the selector accepts it if the user has a sufficient role, and denies it otherwise.

However, in general, the decision for a single action can change according to the actions previously executed by the target. For instance, in the Chinese Wall model [9], a user can *a priori* access a resource from any company, unless she has accessed in the past a resource from another company in the same conflict-of-interest class. We therefore specify a selector as function $F : Act^* \times Act \to \mathcal{D}$, such that $F(\sigma, a)$ stands for the decision made by the selector F for the action a knowing that the past trace is σ.

Clearly, making a security decision is not enough, this decision must also be enforced. There are many different ways to implement such enforcement, that usually depend on the constraints of the concrete system: process algebra, security automaton, turing machine, etc. For instance, given the set

$\mathcal{D}_{edit} = \{acc, sup, ins(b), stop\}$, a selector $F : Act^* \times Act \to \mathcal{D}$, and a past trace σ we can define the security controller $E^{F,\sigma}$ as:

$$E^{F,\sigma} = \sum_{a \in Act} \begin{cases} a.E^{F,a.\sigma} & \text{if } F(\sigma, a) = acc \\ -a.E^{F,a.\sigma} & \text{if } F(\sigma, a) = sup \\ +a.b.E^{F,a.\sigma} & \text{if } F(\sigma, a) = ins(b) \\ \mathbf{0} & \text{otherwise} \end{cases}$$

In general, an important aspect of a security controller is the global impact it has over the entire trace input by the target. Hence, instead of characterizing all possible enforcement mechanisms, we only consider that there exists a semantical function $\mathbf{sem} : Act^* \times \mathcal{D}^* \to Act^*$, such that $\mathbf{sem}(\sigma, d_1 d_2 \dots d_k)$ stands for the trace σ over which each decision d_i has been sequentially enforced. For instance, given the set $\mathcal{D} = \{acc, stop\}$, we could define:

$$\begin{aligned} \mathbf{sem}(\epsilon, \tau) &= \epsilon \\ \mathbf{sem}(\sigma, \epsilon) &= \epsilon \\ \mathbf{sem}(a.\sigma, d.\tau) &= \begin{cases} a.\mathbf{sem}(\sigma, \tau) & \text{if } d = acc \\ \epsilon & \text{otherwise.} \end{cases} \end{aligned}$$

Finally, given a set $\mathcal{D}$, a selector F and a semantics function **sem**, we can define the abstract monitor corresponding to this selector (we assume that $\mathcal{D}$ and **sem** are fixed for a given context), that is, the function $M_F : Act^* \to Act^*$ defined by first obtaining the decision trace for σ according to F and then by applying the semantical function **sem** on this decision trace.

4.2 Inexact Enforcement

As we said in the Introduction, traditional, qualitative enforcement consists in finding a "good" enforcement mechanism, whenever possible. In this section, we try to address the problem of when it is not possible to find a good enforcement mechanism, and therefore the next best choice must be taken.

In order to quantify the non-correctness of a monitor, given a monitor M and a property ϕ, we have introduced in [15] the set $\mathbf{C}\langle M,\phi\rangle$, which represents all traces for which the monitor outputs a non-secure trace. Similarly, the quantification of the non-transparency is done using the set $\mathbf{T}\langle M,\phi\rangle$, which represents the secure traces that the monitor does not output as they are. More formally, we have:

$$\mathbf{C}\langle M,\phi\rangle = \{\sigma \in Act^* \mid \neg\phi(M(\sigma))\} \qquad \mathbf{T}\langle M,\phi\rangle = \{\sigma \in Act^* \mid \phi(\sigma) \wedge M(\sigma) \neq \sigma\}.$$

When M enforces exactly (i.e., soundly and transparently) ϕ, we clearly have $\mathbf{C}\langle M,\phi\rangle = \mathbf{T}\langle M,\phi\rangle = \emptyset$. Conversely, since an n-safety property cannot be enforced exactly using a selector, at least one of these two sets is necessarily non empty.

These two sets enable us to compare two monitors: we can say that a monitor M_1 is better than a monitor M_2 for a property ϕ if $\mathbf{C}\langle M_1,\phi\rangle \subseteq \mathbf{C}\langle M_2,\phi\rangle$ and $\mathbf{T}\langle M_1,\phi\rangle \subseteq \mathbf{T}\langle M_2,\phi\rangle$. However, there are clearly many incomparable monitors,

and more complex criteria can be used in order to define an ordering relationship over monitors, such as the probability and/or the utility of each trace, the cost of enforcement, etc.

4.3 Probabilistic Future

A runtime enforcement mechanism makes decisions at runtime and therefore should not have any knowledge about the future behavior of the target. This is a particularly important problem to enforce non-safety property: if executing an action requires the *a posteriori* fulfilment of an obligation, and if the monitor cannot be sure that the obligation will be indeed fulfilled, then accepting the action might lead to stay in a non-secure state.

However, in some systems, the controller can use some predictive information about the future, usually by analyzing past behavior of users. A typical example of such information is that of *trust* [25,18]: if we know that the user is trusted to fulfil the obligation with a high probability, then the trace selector might choose to output the action, even though the current output would violate the policy.

We have proposed in [15] the concept of n-selector, which extend that of selector by taking, in addition to the past trace and the current action, the next n steps of the trace. In other words, an n-selector indicates which decision to make when the future is also known. Clearly, it is in general unrealistic to expect to have the exact n future steps, however, in some cases, we can have a probability distribution over the future traces.

It becomes then possible to define a selector in the following way: for each possible future trace of length n, we apply the n-selector, and we weight the returned decision by the probability of that trace; the final decision process then consists in composing all the weighted decision. This composition could be done following traditional access control decision composition (e.g., [31,10]), for instance by including quantitative logical aspects, such as those provided by Subjective Logic [17]. An interesting approach in this regard is that of dealing with fuzzy decisions [12].

4.4 Cost of Enforcement

A characteristic aspect of qualitative enforcement is that it only focuses on the correctness of a trace, and whether it is possible or not to enforce correctly a policy. A typical consequence of this approach is that when the target tries to violate the security policy, then the policy does not specify what are the best actions in order to "fix" the trace. This is particularly true for systems with an expressive editing power, such as insertion or suppression. A recent approach [8] tries to address this question by studying the difference between the input and the output trace, and, roughly speaking, states that if two input traces are similar, then the two corresponding output traces must also be similar.

Another way to address the problem of finding the best editing strategy is to associate each trace modification with a *cost*. For instance, inserting a new action can be associated with the cost of creating such operation, deleting an action can be associated with the absence of gain of executing this action, stopping the system can be associated with a loss of usability, etc.

Hence, given a cost domain (e.g., the set of positive real numbers $\mathbb{R}^+$), we associate each decision and each action with a cost. The cost of editing a whole trace is then simply the sum of all atomic costs.

We presented such as cost model in [14], which allowed us to specify the cost associated with a monitor, and in particular to set a *cost limit* that a monitor is not allowed to cross. It is therefore possible to specify the best editing strategy by associating particular costs with some actions. For instance, by setting the acceptance cost to a minimum, we showed that it is always the best strategy to accept a correct action, which means that transparency can be obtained as a side-effect of a cost model. Similarly, by associating an infinite cost with the suppression cost of a particular action, we can model the concept of *uncontrollable action* [5], that is, an action that has to be accepted, such as the tick of a clock.

Finally, defining a cost model leads to the definition of a partial ordering over monitors: a monitor M_1 is "better" than a monitor M_2 if, and only if, the cost of M_1 for each trace is less than the cost of M_2. Furthermore, by considering a probability distribution over traces, such as described in the previous section, we have provided a way to calculate the expected cost of a monitor, by weighting the cost of each trace by its probability.

4.5 Trace Reward/Utility

The cost model presented in the previous section characterizes the idea that each atomic action over a trace comes with a cost. Following several recent approaches [19,29,27], it is also possible to adopt a more global point of view, where we consider the utility, positive or negative, of a whole trace. This trace utility can be given by a function $Q : Act^* \to \mathbb{V}$, where $\mathbb{V}$ is a value domain.

Note that the main intuitive difference between the cost domain of the previous section and such value domain is that the cost domain is expected to be only positive (a negative cost is somehow counter-intuitive), while a value can be either positive or negative. For instance, a non-secure trace can be associated with a negative value (for instance, the fine of violating the policy), while a secure trace can be associated with a positive value (the gain of the system when executing this trace).

In this setting, defining the best monitor consists in optimizing the value of the overall system. We have modelled in [27] an access control mechanism as a Markov Decision Process, where the utility value is defined through the reward function, associated with each state of the system. We have defined for each decision of the monitor its semantical effect over the state, thus allowing the system to know exactly which state can be reached by which actions. Finally, the optimal decision process can be specified, and can take the probability distribution of traces into account.

5 Related Work

In this section we present some of the related work on controller theory and security enforcement. Starting from Schneider's seminal work [32], Ligatti et al. have defined in [6,7] four different kind of security automata which deal with

finite sequences of actions: truncation, suppression, insertion and edit automata. A modelling of these security automata through process algebra operators is presented in [26], enabling the compositionality of the approach with the usage of partial model checking and allowing for the application of existing results on process algebras to the analysis, verification and synthesis of secure systems.

In [4] a mixed approach to access control is proposed to enforce both safety and liveness properties: security critical code is enclosed in policy framings, in particular safety framings and liveness framings, which enforce respectively safety and liveness properties of execution histories. This is however a static analysis that over-approximates behavior history expressions. This approach efficiently combines static analysis and run-time checking: a program with policy framings is compiled into an equivalent one without framings, but instrumented with local checks. The static analysis determines which checks are needed and where they must be inserted to obtain a program respecting the given security requirements. The execution monitor is essentially a finite-state automaton associated with the relevant security policies.

In [13], the authors present the enforcement strategy of Gate Automata. It is able not only to halt the execution of the target if something goes wrong but is also able to add and suppress actions for correcting the target behavior when possible according to some trust measure whose management is integrated into the enforcement strategy.

In all previous works, the synthesis problem is not addressed. The synthesis of controllers is a framework addressed also in other research areas (e.g. [3,35]). Many other approaches to the controller synthesize problem are based on game theory (e.g. [2,20,21]). Indeed, different kinds of automata are used to model properties that must be enforced. Games are defined on the automata in order to find the structure able to satisfy the given properties.

6 Conclusion and Future Directions

We have presented in this paper two different aspects of runtime enforcement of security policies: how to *qualitatively* synthesize a controller, and a range of *quantitative* properties can be expressed over a controller. The next logical step is therefore to study how to synthesize a controller in a quantitative way, that is, how to ensure that the synthesized controller will respect some quantitative properties. A promising way to do so is to adopt partial model checking techniques for probabilistic process algebras and temporal logics, in order to integrate quantitative elements.

Another interesting lead is to reconsider traditional approaches about the definition of what kind of policies can be enforced in a given context by also including some notions of costs, precision, etc. For instance, we could wish to distinguish policies that can only be enforced with an infinite cost from those that can be enforced with a finite cost. Similarly, it might be useful to define the class of policies for which the enforcement would cost more than the gain provided by such enforcement.

Acknowledgments. This work has been partially supported by the EU projects NESSoS and CONNECT.

References

1. Andersen, H.R.: Partial model checking. In: LICS 1995: Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science, p. 398. IEEE Computer Society (1995)
2. Arnold, A., Vincent, A., Walukiewicz, I.: Games for synthesis of controllers with partial observation. Theoretical Computer Science 303(1), 7–34 (2003)
3. Badouel, E., Caillaud, B., Darondeau, P.: Distributing finite automata through petri net synthesis. Journal on Formal Aspects of Computing 13, 447–470 (2002)
4. Bartoletti, M., Degano, P., Ferrari, G.-L.: Checking Risky Events Is Enough for Local Policies. In: Coppo, M., Lodi, E., Pinna, G.M. (eds.) ICTCS 2005. LNCS, vol. 3701, pp. 97–112. Springer, Heidelberg (2005)
5. Basin, D., Jugé, V., Klaedtke, F., Zălinescu, E.: Enforceable Security Policies Revisited. In: Degano, P., Guttman, J.D. (eds.) Principles of Security and Trust. LNCS, vol. 7215, pp. 309–328. Springer, Heidelberg (2012)
6. Bauer, L., Ligatti, J., Walker, D.: More enforceable security policies. In: Proceedings of the FLoC 2002 Workshop on Foundations of Computer Security, July 25-26, pp. 95–104. DIKU Technical Report, Copenhagen (2002)
7. Bauer, L., Ligatti, J., Walker, D.: Edit automata: Enforcement mechanisms for run-time security policies. International Journal of Information Security 4(1-2) (2005)
8. Bielova, N., Massacci, F.: Predictability of Enforcement. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) ESSoS 2011. LNCS, vol. 6542, pp. 73–86. Springer, Heidelberg (2011)
9. Brewer, D.F.C., Nash, M.J.: The Chinese Wall Security Policy. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 329–339 (May 1989)
10. Bruns, G., Huth, M.: Access-control policies via Belnap logic: Effective and efficient composition and analysis. In: Proceedings of the CSF 2008, pp. 163–176. IEEE Computer Society (2008)
11. Bruns, G., Sutherland, I.: Model Checking and Fault Tolerance. In: Johnson, M. (ed.) AMAST 1997. LNCS, vol. 1349, pp. 45–59. Springer, Heidelberg (1997)
12. Cheng, P.-C., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M., Reninger, A.S.: Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. In: Proceedings of Security and Privacy 2007, pp. 222–230. IEEE (2007)
13. Costa, G., Matteucci, I.: Gate automata-driven run-time enforcement. Computers & Mathematics with Applications 63(2), 518–524 (2012)
14. Drábik, P., Martinelli, F., Morisset, C.: Cost-aware runtime enforcement of security policies. In: Proceedings of the Security and Trust Management 2012. LNCS (to appear, 2012)
15. Drábik, P., Martinelli, F., Morisset, C.: A quantitative approach for the inexact enforcement of security policies. In: Proceedings of the Information Security Conference 2012. LNCS (to appear, 2012)
16. Ferraiolo, D.F., Kuhn, D.R.: Role-based access control. In: Proceedings of the 15th National Computer Security Conference, pp. 554–563 (1992)
17. Jøsang, A.: Conditional reasoning with subjective logic. Multiple-Valued Logic and Soft Computing 15(1), 5–38 (2009)

18. Krautsevich, L., Lazouski, A., Martinelli, F., Mori, P., Yautsiukhin, A.: Usage Control, Risk and Trust. In: Katsikas, S., Lopez, J., Soriano, M. (eds.) TrustBus 2010. LNCS, vol. 6264, pp. 1–12. Springer, Heidelberg (2010)
19. Krautsevich, L., Martinelli, F., Morisset, C., Yautsiukhin, A.: Risk-Based Auto-delegation for Probabilistic Availability. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cuppens-Boulahia, N., de Capitani di Vimercati, S. (eds.) DPM 2011 and SETOP 2011. LNCS, vol. 7122, pp. 206–220. Springer, Heidelberg (2012)
20. Kupferman, O., Madhusudan, P., Thiagarajan, P.S., Vardi, M.Y.: Open Systems in Reactive Environments: Control and Synthesis. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, p. 92. Springer, Heidelberg (2000), citeseer.ist.psu.edu/kupferman00open.html
21. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems (extended abstract), http://citeseer.ist.psu.edu/302111.html
22. Martinelli, F.: Analysis of security protocols as *open* systems. Theoretical Computer Science 290(1), 1057–1106 (2003)
23. Martinelli, F., Matteucci, I.: A framework for automatic generation of security controller. STVR Journal (2010)
24. Martinelli, F., Matteucci, I.: Partial model checking, process algebra operators and satisfiability procedures for (automatically) enforcing security properties. Technical report, IIT-CNR. Presented at the International Workshop on Foundations of Computer Security, FCS 2005 (2005)
25. Martinelli, F.: Towards an Integrated Formal Analysis for Security and Trust. In: Steffen, M., Zavattaro, G. (eds.) FMOODS 2005. LNCS, vol. 3535, pp. 115–130. Springer, Heidelberg (2005)
26. Martinelli, F., Matteucci, I.: Through modeling to synthesis of security automata. Electr. Notes Theor. Comput. Sci. 179, 31–46 (2007)
27. Martinelli, F., Morisset, C.: Quantitative access control with partially-observable markov decision processes. In: Proceedings of CODASPY 2012, pp. 169–180. ACM, New York (2012), http://doi.acm.org/10.1145/2133601.2133623
28. Merlin, P., Bochmann, G.V.: On the Construction of Submodule Specification and Communication Protocols. ACM Transactions on Programming Languages and Systems 5, 1–25 (1983)
29. Molloy, I., Dickens, L., Morisset, C., Cheng, P.-C., Lobo, J., Russo, A.: Risk-based security decisions under uncertainty. In: Proceedings of CODASPY 2012, pp. 157–168. ACM, New York (2012), http://doi.acm.org/10.1145/2133601.2133622
30. Moses, T.: eXtensible Access Control Markup Language TC v2.0 (XACML) (February 2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
31. Ni, Q., Bertino, E., Lobo, J.: D-algebra for composing access control policy decisions. In: Li, W., Susilo, W., Tupakula, U.K., Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS, pp. 298–309. ACM (2009)
32. Schneider, F.B.: Enforceable security policies. ACM Transactions on Information and System Security 3(1), 30–50 (2000)
33. Street, R.S., Emerson, E.A.: An automata theoretic procedure for the propositional μ-calculus. Information and Computation 81(3), 249–264 (1989)
34. Walukiewicz, I.: A Complete Deductive System for the μ-Calculus. Ph.D. thesis, Institute of Informatics, Warsaw University (June 1993)
35. Wong-Toi, H., Dill, D.L.: Synthesizing Processes and Schedulers from Temporal Specifications. In: Clarke, E., Kurshan, R.P. (eds.) CAV 1990. LNCS, vol. 531, pp. 272–281. Springer, Heidelberg (1991)

Design and Implementation of a Cloud-Based Assured Information Sharing System

Tyrone Cadenhead*, Murat Kantarcioglu,
Vaibhav Khadilkar, and Bhavani Thuraisingham

Department of Computer Science,
The University of Texas at Dallas, Richardson, TX 75083
{thc071000,muratk,vvk072000,bxt043000}@utdallas.edu
http://www.utdallas.edu

Abstract. The advent of cloud computing and the continuing movement toward software as a service (SaaS) paradigms have posed an increasing need for assured information sharing (AIS) as a service in the cloud. This paper describes the first of its kind assured information sharing system that operates in a cloud. The idea is for each organization to store their data and the information sharing policies in a cloud. The information is shared according to the policies. We describe a cloud-based information sharing framework that utilizes Semantic Web technologies; our framework consists of a policy engine that reasons about the policies for information sharing purposes and a secure data engine that stores and queries data in the cloud. We also describe the operation of our system with example policies.

Keywords: Assured Information Sharing, Cloud Computing, Resource Description Framework, Policies.

1 Introduction

The cloud computing paradigm enables the sharing of large amounts of data securely and efficiently. Furthermore, the advent of cloud computing and the continuing movement toward software as a service (SaaS) paradigms have posed an increasing need for assured information sharing (AIS) as a service in the cloud. In order to satisfy the cloud-centric assured information sharing (AIS) needs of coalition organizations, there is a critical need to develop an AIS framework that operates in the cloud. To our knowledge, no such system currently exists. In an earlier paper [1], we described the design of a system called CAISS: a Cloud-centric Assured Information Sharing System (CAISS) that utilizes the technology components we have designed in-house as well as open source tools. CAISS consists of two components: a cloud-centric policy manager that enforces policies specified in RDF (resource description framework) [2] and a cloud-centric data

* This material is based upon work supported by The Air Force Office of Scientific Research under Award No. FA-9550-08-1-0260. We thank Dr. Robert Herklotz for his support.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 36–50, 2012.

manager that will store and manage data also specified in RDF. This RDF data manager is essentially a query engine for SPARQL (SPARQL Protocol and RDF Query Language) [4], a language widely used by the Semantic Web community to query RDF data. RDF is a Semantic Web language that is considerably more expressive than XML- (extensible Markup Language) based policy languages or specifying and reasoning about policies. Furthermore, our policy manager and data manager will have seamless integration since they both manage RDF data.

While many of the ideas in our previous paper were in the concept stages, in this paper, we describe the detailed design and implementation of CAISS. We have developed a comprehensive AIS framework that seamlessly operates in the cloud. Our framework contains a three layer architecture that consists of a user interface layer, a policy engine layer and a data connection layer that integrates multiple data sources in the cloud. To our knowledge this is the first of its kind AIS framework that operates in the cloud. We describe the detailed design and implementation of our system in Section 2. In particular, the system architecture, operations, modules and usage are discussed. We then provide a description of the novel features of our implementation in section 3. The paper is concluded in Section 4 with a discussion of future work.

2 Architecture

Our system architecture consists of three layers (see Fig. 1). At the front-end, we have a user interface; the middle layer consists of our policy engine logic; and at the backend, we have our data stores. We will first give an overview of the configuration of our framework. Then we will define each of the layers in our architecture.

2.1 RDF Framework Configuration

Our policy engine framework is driven by RDF configuration documents (see Fig. 2), which encode the logic of the user interface layouts and customizable parameters, the policy engines and their usage, and the mappings of dereferenceable uniform resource identifiers(URI) to the data stores using the available data connections. Our policy engine framework can be used as a key enabler in augmenting security for RDBMS's, as well as cloud-based systems. RDBMS's are developed with atomicity, concurrency and durability in mind, but are normally shipped with limited support for access control. A cloud storage layer allows the agencies to store and scale policies with finer levels of control over RDF resources. The cloud was developed with scalability and availability in mind, but security considerations were neglected. Our policy engine can be configured to complement policies in a RDBMS system with an entry point for supporting security policies over cloud-based back-ends.

A loosely coupled system also provides easy configuration and flexibility to our RDF policy engine framework. Each component is abstracted from the others by employing RDF documents consisting of an agency's preferences for a

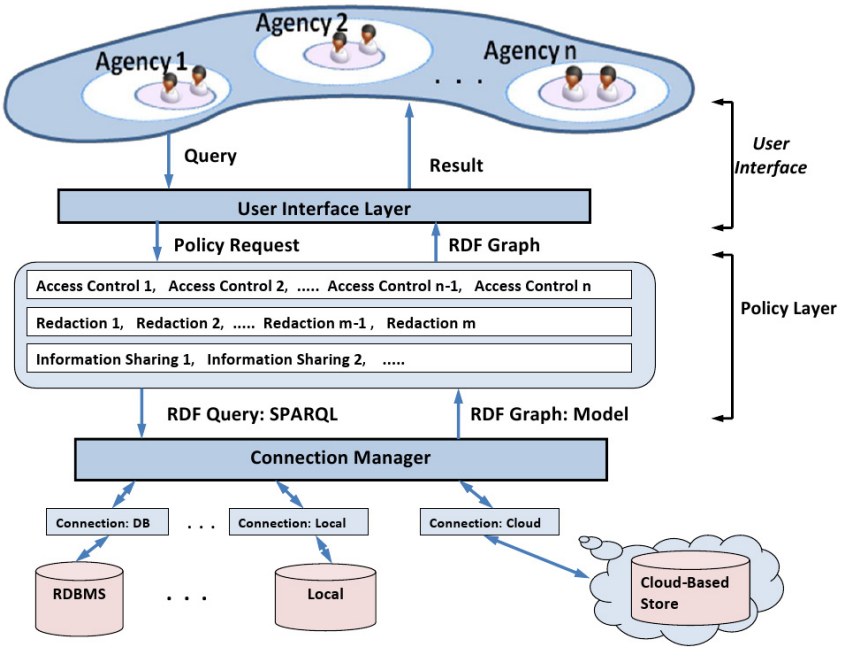

Fig. 1. Architecture

policy or data connection to a data store. Furthermore, a loosely coupled web front-end promotes easier maintenance and reusability of the policy framework, since an adapter pattern abstracts the mapping of the web interfaces (and communications) to the other layers. An abstraction hides the actual implementation and intricacies of the policy engine manager and data managers from the agencies. This therefore allows agencies to specify their policies in any representation languages, such as XML, RDF or Rei [3]; an adapter hides the translation of high-level policy specification to policy implementation.

2.2 User Interface Layer

To enable a one-to-one interaction between a user and our policy framework, a web-based user interface is built on top of the policy layer. Rich client and open source web technologies simplify the interactions between users, web pages and the underlying policy and data layers. This integration has many advantages. The policy framework operates in a distributed environment and has a greater geographical spread; therefore, agencies and users have mobility. The web interface requires users to create an account (also a registration) and choose unique credentials, which will then be used by the users to identify them to the policy framework. A form-based authentication pattern, as well as a challenge-response test distinguishes legitimate users from robots (which may pose as normal users). The legitimate users are presented with a querying screen that

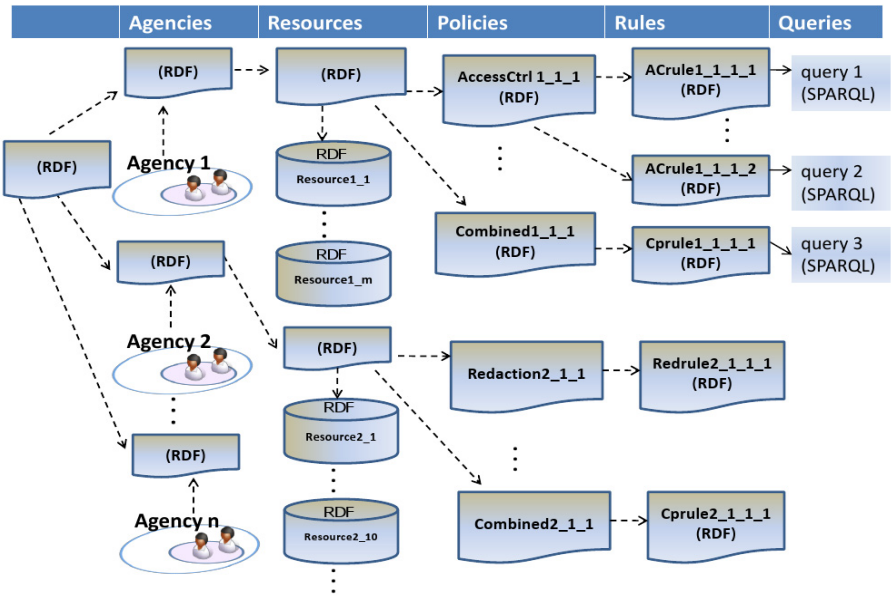

Fig. 2. Framework Overview

allows them to compose SPARQL queries once they have been authenticated. Note that SPARQL [4] is a query language for RDF and is used for retrieving data from triple stores. The input SPARQL queries are validated at the user interface layer and then sent to our policy engine layer, which in turn returns a resultant RDF graph that is then displayed on a web page.

User Registration

The User Registration presents the user the opportunity to register with the system using a web registration form. The registration form captures the user's name, password and other metadata about the user. Metadata could be an agency that the user is a part of, or data that is used for mapping the user's credentials to a role, which is to be performed by the user.

The following RDF graph displays contents from a user configuration file. The final triple in the RDF graph contains a dereferenceable URI to another RDF graph, which then contains a list of dereferenceable URIs of the actual resources that the user is allowed to query.

```
# users
<http://policy.org/agency/pol#users>
      pol:user    <http://policy.org/agency/pol#user1> .

# user details and resources
<http://policy.org/agency/pol#user1>
```

```
pol:name "user1" ;
pol:passwd "_:b1" ;
pol:organization  <http://policy.org/agency/pol#Agency1> ;
pol:resourcelist  <http://example/users/resources/user1> .
```

Agency Registration
The Agency Registration comprises a sequence of web pages, each being a child page of the previous one. The process commences with an agency registering information to describe itself. First, an agency registers important metadata about itself. This metadata is captured as an RDF document, which can be used to introduce one agency to another, and therefore, should be self-describing. Some example triples in this metadata could assert an agency's name, address, industry, affiliations, etc. Second, an agency records its resources. A RDF resource has a unique URI, which is a dereferenceable URI to an agency's RDF resources at a datastore, which contains both the sensitive and non-sensitive data for the agency; this is the information that is normally stored in a relational database, but is now migrated to the cloud. Third, an agency defines the policies for its resources. An agency may choose among the various policies that are supported at the policy engine layer. Examples of policies are access control, redaction, information sharing, etc. Fourth, an agency describes various policy rules for a policy. Note that an agency may use access control to protect its resources; however, the agency may need more than one rule for a particular policy choice. For example, one access control rule may specify a positive authorization, while another may specify a negative authorization on the same resource. Finally, an agency specifies queries. It is a very popular technique to write policy rules as views (i.e., SPARQL queries) over a data store. An agency may specify in its policy rule configuration document that queries are be materialized or that they be non-materialized. A materialized query may speed up the policy execution, while a non-materialized query refreshes the result set in real-time. Note that there is a correspondence between the web pages at the agency registration and the RDF documents in Fig. 2.

2.3 Policy Engines

An agile environment pushes policy designers to constantly fine-tune or extend their policies to rapidly adapt to ever-changing conditions, thus ensuring that data integrating and data combinations do not violate data confidentiality, especially when quick actions are critical (e.g., in intelligence). To meet this demand, our policy engine layer supports many policy engines, while the cloud supports many policy configuration documents.

The Policy Engine Layer first evaluates the user queries against the stored data resources (which can be traditional data, or provenance metadata). A data resource is characterized by a URI which connects to an actual RDF graph in the data storage layer. The policy layer uses a factory object to create the underlying policies. The factory exposes a policy through a consistent interface, thus making

it easy to extend our policy engine to support other types of policies in the future. We currently support access control, redaction, and information sharing policies. To support traditional policies, we use SPARQL queries to define views over the data resources, where a view can be associated with positive and negative authorizations or a target RDF graph in a subgraph replacement procedure. An important metadata is provenance, which records the history of a piece of data item. However, provenance takes on a directed acyclic graph (DAG) structure [5], and as such requires its own policies. Therefore, we support the use of regular expression SPARQL queries for access control policies [6], as well as redaction policies [7]. We have also implemented information sharing policies over data and provenance that allow cooperating agencies to share information based on mutual agreements [8].

A policy engine takes as input a user's credential and a dereferenceable URI. It then evaluates the underlying logic of a policy before returning a new RDF graph (or model) to the user interface layer. The dereferenceable URI points to a configuration document, which itself contains other dereferenceable URIs to the policies about an agency's resource and to the agency's resource at the data layer. An agency's resource is an RDF document, with triples at one or more classification levels. For example, an entire RDF document would be classified as sensitive in case it contains intelligence information, or some subset of triples may have actual intelligence information. An agency therefore requires more than one type of policy to achieve fine-grain control over its resources. A policy is therefore defined by an interface, which allows the implementation of the logic of each policy. By migrating its policies to the cloud, an agency overcomes the restriction on the number of policy definitions previously possible. The following subsections summarize various policy types. In the subsections below, we discuss the details of the policy engine layer. This layer comprises many policy types, for example, access control, redaction, information sharing, to name a few. We will also motivate the need for a flexible policy engine by discussing each of these policy types in turn.

Access Control Policy Engine

An access control policy authorizes a set of users to perform a set of *actions* on a set of *resources* within an *environment*. Unless authorized through one or more access control policies, users have no access to any resource of the system. There are different kinds of access control policies, which can be grouped into three main classes [9]. These policies differ by the constraints they place on the sets of *users*, *actions* and *objects* (access control models often refer to *resources* as *objects*). These classes are (1) RBAC, which restricts access based on roles; (2) discretionary access control (DAC), which controls access based on the identity of the user; and (3) mandatory access control (MAC), which controls access based on mandated regulations determined by a central authority.

Policies based on RBAC are often used to simplify the management of policy mappings, which is a common feature in the three classes of access control policies. Policy creation and manageability are important in getting finer levels of

access control over the shared resources. We use the convention that a permission is a unique pair of (*action*, *resource*). Given n resources, m users and a set of only two actions (read, write), we have a maximum of $2 \times n$ possible permissions. This gives $m \times (2 \times n) = c_1 n$ mappings. A further improvement of RBAC is the case where there is at least one role with two or more users assigned to it, from a possible set of r roles. Therefore, we have $r \times (2 \times n) = c_2 n$ mappings and we also assume that $c_2 \leq c_1$. However, even with this simplification, the number of policies needed to achieve finer levels of access control in a dynamic and agile community may be intractable. Our cloud-centric policy framework addresses this by providing the agencies the ability to support and scale their access control policies to meet their ever-growing security needs.

Redaction Policy Engine
A redaction policy identifies and removes sensitive information from a document before releasing it to a user. Unlike access control policies, which restrict access, redaction policies encourage sharing of information, by ensuring that sensitive or proprietary information is removed (or obscured) before providing the final RDF graph (referred to as a redacted graph) to a user's query. Redaction policies rely on a transformation operation in order to circumvent any identifying or sensitive information. The redaction policy engines currently supported rely on a graph transformation technique that is based on a graph grammar approach (which is presented in [10,11]. Basically, there are two steps to applying a redaction policy over a directed labeled RDF graph: (i) Identify a resource (or subgraph) in the original RDF graph that we want to protect. This can be done with a graph query (i.e., a query equipped with regular expressions). (ii) Apply a redaction policy to this identified resource in the form of a graph transformation rule. An implementation of this graph transformation is used in [7] for redacting provenance graphs.

Information Sharing Policy Engine
An information sharing policy allows agencies to determine the context in which their resources are shared or combined with resources from other agencies. An information sharing policy engine has logic for processing a query requesting information on two or more RDF graphs simultaneously. We illustrate this using the following SPARQL query.

SELECT ***B*** FROM NAMED uri1 FROM NAMED uri2 WHERE P,

where P is a graph pattern, ***B*** is a tuple of variables appearing in P and uri1 and uri2 are dereferenceable URIs for two resources, R1 and R2. Resources R1 and R2 may be from the same agency, in case an agency strictly requires a partitioning of its resources based on confidentiality concerns or they could belong to two agencies, Agency 1 and Agency 2 respectively. Therefore, each of these resources may define individual information sharing policy rules. We define an operator $\odot$, so that an information sharing policy is now evaluated over uri1 $\odot$ uri2. The operator $\odot$ can be implemented as a graph operation over a RDF graph. Note

that, $\odot$, could be one of the following operators: $\cap$, $\cup$ or $-$ and can also be applied to an original RDF graph or to previous one, which resulted from the operator, $\odot$. In order to execute the operator, $\odot$, we define a graph recursively as follows.

- ϵ is a graph.
- The set of graphs are closed under intersection ($\cap$), union ($\cup$) and set difference ($-$). Let G_1 and G_2 be two graphs, then $G_1 \cup G_2$, $G_1 \cap G_2$ and $G_1 - G_2$ are graphs, such that if $t \in G_1 \cup G_2$ then $t \in G_1$ or $t \in G_2$; if $t \in G_1 \cap G_2$ then $t \in G_1$ and $t \in G_2$; or if $t \in G_1 - G_2$ then $t \in G_1$ and $t \notin G_2$.

The following RDF graph lists triples from a combined policy configuration document containing policies with embedded logic for sharing two resources, R1 and R2, which belong to two agencies, Agency 1 and Agency 2 respectively.

```
# entity
<http://policy.org/entity/pol#Combined1_1_1>
   pol:owner <http://policy.org/entity/pol#Agency1>;
      pol:rule  <http://policy.org/entity/pol#Cprule1_1_1_1> .

# mappings
<http://policy.org/entity/pol#Cprule1_1_1_1>
   pol:agency<http://policy.org/entity/pol#Agency2>;
      pol:operator    "UNION" ;
      pol:type   "combined1" .
```

This policy works at the level of the agencies. For example, Agency 1 shares all its resources as a union with all of Agency 2 resources. The policy type allows an agency to have modes of sharing. For example, a type *combined*1 provides sharing at the agency level, while another policy type, *combined*2, could offer a finer level of control in determining how Agency 1 shares each of its resources with a classification of a resource for Agency 2. In other words, information sharing policies can incorporate contextual information about an agency and metadata about each of its resources at the resource level. The following shows two policy types for our information sharing policies:

1. **combined1** $\forall r1 \in Agency1, \forall r2 \in Agency2$, use r1$\cup$r2. This policy states that Agency 1 shares all its resources with Agency 2 as a union of the resources.
2. **combined2** let $r1_1, r1_2, \ldots, r1_n \in Agency1$, use $r1_1 \cup r2$, $r1_2 \cap r2$, $\forall r2 \in Agency2$. This policy offers a finer level of control.

Provenance Policy Engines

Sometimes the relationships among the triples in an RDF graph need be taken into consideration, when defining policies. The three policy types discussed so far fail to address the cases where sensitive information is implicit in the various paths within an RDF graph. We will explore other policy engines in this section.

The focus will be on the definition of policy engines tailored to the execution of access control and redaction policies over a provenance graph, but is applicable to information policies as well. We will base the logic of these policy engines on [6], which discusses an access control policy language for provenance and [7], which discusses how to perform redaction over provenance. We will first give an example of a provenance graph and the type of provenance information which may exist in an example provenance graph. Then we will present brief definitions of some of the theory behind executing policies over a provenance graph.

Fig. 3 shows an intelligence example as a provenance graph using a RDF representation that outlines a flow of a document through a server located in some unfriendly territory (or at another agency posing a potential threat). This document was given to a journalist. Also, the final report can be traced back to a CIA agent. The contents of this provenance graph could serve to evaluate the trustworthiness of the servers (i.e., processes in the example graph) from which the document originated. This example provenance graph also shows the base skeleton of the actual provenance, which is usually annotated with RDF triples indicating contextual information, e.g., time and location. Note that a RDF graph is composed of triples $(subject, predicate, object)$. In this example, the predicates (i.e., arcs) are labeled with the OPM abstract predicate [12] labels and these predicates make up a vocabulary.

The information embedded in the graph in Fig. 3 represents a directed RDF graph. A provenance path in Fig. 3 is defined as follows:

Definition 1. *(Provenance Path) Given a provenance graph, a provenance path* $(s\ \rho\ o)$ *is a path* $s(\xrightarrow{\rho})o$ *that is defined over the provenance vocabulary* V *using regular expressions.*

Definition 2. *(Regular Expressions) Let* Σ *be an alphabet of terms in* V*, then the set* $RE(\Sigma)$ *of regular expressions is inductively defined by:*

- $\forall x \in \Sigma, x \in RE(\Sigma)$;
- $\Sigma \in RE(\Sigma)$;
- $\epsilon \in RE(\Sigma)$;
- *If* $A \in RE(\Sigma)$ *and* $B \in RE(\Sigma)$ *then:*
  $A|B, A/B, A^*, A^+, A? \in RE(\Sigma)$.

The symbols | and / are interpreted as logical OR and composition respectively.

Our intention is to define paths between two nodes by edges equipped with * for paths of arbitrary length, including length 0 or + for paths that have at least length 1. Therefore, for two nodes x, y and predicate name p, $x(\xrightarrow{p})^*y$ and $x(\xrightarrow{p})^+y$ are paths in G.

A SPARQL query extended with regular expressions [13] can define a resource (or subgraph) of the provenance graph in Fig. 3 as follows:

Example 1. (**Provenance Path Query**)

```
Select ?x
{ ex:PubRpt1 arq:OnPath("([opm:WasGeneratedBy]/
       [opm:WasTriggeredBy]/[ex:location])" ?x). }
```

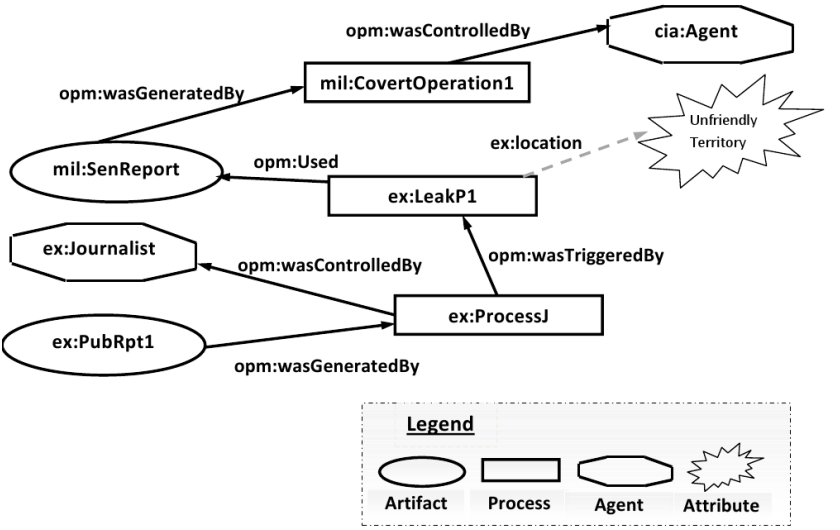

Fig. 3. Provenance Graph

This query would return the location as a binding to the variable x and could be used to pinpoint the origin of a compromise (and leakage) of the original report. This could also serve to alert policy designers to add appropriate policies for reports and servers in their respective agencies.

Policy Sequence

The execution of the policies over an agency's resource results in a policy sequence. In particular, a protected resource could employ the services of multiple policy engines and policy types. Each policy type produces a new subgraph of its input RDF graph. It is important to note that the effect of a policy is directly dependent on the RDF graph it receives as input, and furthermore, the effect may be different from the original effect the policy was intended to achieve. A sequence takes the original input graph through a series of transformations until a final RDF graph is returned to the user. Note that the success of a policy rule (which is implemented as a SPARQL query) returning a particular set of RDF triples is dependent on the transformation step at which the rule was applied in a policy sequence. We illustrate this using the following SPARQL query:

CONSTRUCT G WHERE P,

G is a newly constructed graph, which contains a set of triples that satisfy condition P in the input graph. A policy protecting the following RDF triples,

```
<http://cs.utdallas.edu/semanticweb/Prov-AC/agency#Agent_1>
     foaf:name "John Brown";
     foaf:projectHomepage <http://www.agency1.gov/> .
```

will fail if either the name or project home page triple was earlier removed or altered by a previous policy rule. A policy precedence feature in the framework helps an agency determine the ordering of its policies. In the user interface layer, an agency configures the ordering of its policies. The policy sequence is then stored in a RDF sequence file (using the "rdf:seq" feature of the RDF specification). When a query is evaluated, the policy framework will in turn invoke each policy in the intended order.

Rule Sequence
In a similar way, a policy may be implemented using a set of rules. For example, to fully redact a shared resource, an agency may need a separate rule to redact each sensitive triple in an RDF graph. Each rule is triggered when a triple (or set of triples) meet some specified criteria in the input graph. Note that each rule transforms the current state of a shared resource. Therefore, each sequencing of the rules will impact the final graph.

2.4 Data Layer

At the Data Layer is a connection factory, which acts as a facade, for creating connection objects. These connection objects expose the same properties (functionally) as public methods to the policy designer. This makes it easier for the policy designer to concentrate on the policy engine design. The policy designer makes a call to an RDF Policy Factory, which returns an RDF model object. This RDF model object is backed by a connection store, which can be a local connection, a relational database connection or a cloud connection. During the registration process, an agency is given an opportunity to decide where it wants to store its resources and configuration documents. It is recommended that the smaller configuration documents be stored locally on disk (or in a local database) to enable quick access to them. Local connections also consume lower bandwidth, offer real-time access and enable development before deployment. However, an agency may decide to store them in a private cloud (or on a remote database server) to take advantage of the added protection there.

The connection factory also enables agencies to store their resources in any cloud infrastructure. For example an agency's resources could reside in a private cloud, or a community cloud or a public cloud. A private cloud deployment provides more control, in that agencies could house their own cloud. A community cloud is provisioned for exclusive access by a specific community, thus serving the common interest of cooperating agencies. A public cloud is open to the public and thus susceptible to more vulnerability due to the loss of control over the data uploaded onto the public cloud. Agencies may choose to use a mixture of connections and also employ more than one deployment simultaneously (e.g., a hybrid cloud model).

3 Features of Our Policy Engine Framework

In the subsections below, we present some novel features of our policy engine framework.

3.1 Policy Reciprocity

Policy Reciprocity enables agencies to specify policies when knowledge of the other agencies, their resources or policy specifications are available. This is made possible via the registration process, where agencies make metadata available about themselves, their resources and associated policies. The following discussion provides scenarios for policy reciprocity.

*Agency*1 wishes to share its resources if *Agency*2 also shares its resources with it. Current access control and redaction policies do not provide for this reciprocity. Our framework provides information sharing policies, which allow agents to define policies based on reciprocity and mutual interest amongst cooperating agencies.

We present two sample information sharing policies below:

1. $\forall r1 \in Agency1, \forall r2 \in Agency2$, use r1$\cup$r2.
   This policy states that *Agency*1 shares all its resources with any resource of *Agency*2 as a union of the resources (i.e., $\odot \in \{\cup\}$).
2. let $r1_1, r1_2, \ldots, r1_n \in Agency1$, use $r1_1 \cup r2$, $r1_2 \cap r2$, $\forall r2 \in Agency2$.
   This policy offers a finer level of control and defines the combined operator, $\odot \in \{\cap, \cup\}$.

Conditional Policies
A consequence of policy reciprocity is allowing the use of conditional sharing policies. For example, *Agency*1 shares its resources with *Agency*2 if *Agency*2 does not share *Agency*1's resources with Agency3. We present a sample information sharing policy below:

1. $\forall r1 \in Agency1, \forall r2 \in Agency2$, *Agency*1 defines r1$\cap$r2. If $\forall r3 \in Agency3$, then
   - *Agency*2 does not define any sharing policy of the form r1$\cap$r3,
   - or *Agency*2 does not define any sharing policy of the form r1 $\subseteq$ r2 $\odot$r3, where $\odot \in \{\cup, \cap\}$.

Policy Symmetry
Another consequence of policy reciprocity is to have symmetry in the sharing of policies. For example, *Agency*1 shares its resources with *Agency*2 with a combined operator, $\odot$, if *Agency*2 also shares its resources with *Agency*1 using the same combined operator, $\odot$. We present a sample information sharing policy below:

1. $\forall r1 \in Agency1, \forall r2 \in Agency2$, *Agency*1 uses r1$\cup$r2 if *Agency*2 also uses r2$\cup$r1.

3.2 Develop and Scale Policies

To enable freedom of maneuverability across the information environment and to deliver the power of information to ensure mission success, an agency should

be able to rapidly develop policies and deploy them as needed. We next discuss the features that are available to an agency during and after development of its policies.

Policy Development
*Agency*1 wishes to simulate a live environment and create test scenarios to visualize the results of each policy configuration. Our policy framework provides three configurations: (i) a standalone version for development and testing; (ii) a version backed by a relational database; and (iii) a cloud-based version that achieves high availability and scalability while maintaining low setup and operation costs.

Sequencing Effects
*Agency*1 wishes to vary the result set to a user's query based on the user's credentials. The policy sequence feature can be used to configure different outcomes by permuting the policies and their respective rules.

Rapid Elasticity
*Agency*1 identifies recent security vulnerabilities in its existing policy configurations and wishes to extend (or grow) its existing policy set with support for policies at a finer granularity. Our policy engine provides a policy interface that should be implemented by all policies; therefore, we can add newer types of policies as needed. In addition, our policy engine gives an agency rapid elasticity, whereby the capabilities available by our policy framework appear unlimited.

Location Independence
*Agency*1 wishes to store its resources closer to where it is consumed, but with little or no change at the policy layer. Our policy engine provides location independence whereby the policy engine has no control or knowledge over the exact location of the resources, but may be able to access the resources through a specified location using the connection manager. Note that an agency's resources can be in any cloud, geographically. The ability to locate any resource by a dereferenceable URI offers much flexibility.

Deployment Models
*Agency*1 can take advantage of different deployment models. For example, a private cloud, a hybrid cloud, a community or a public cloud. The connection manager allows an agency to choose among a list of connection types based on different risk factors and objectives in protecting its data confidentiality.

3.3 Justification of Resources

Provenance makes available an explanation about why information was manipulated and a trace to the source of the information manipulation. This establishes trust among agencies, thus facilitating partnerships for common goals.

*Agency*1 asks *Agency*2 for a justification of resource R2. The current commercial access control policies are mainly designed to protect single data items, while current redaction policies are designed for redacting text and images. Our policy engine allows agents to define policies over provenance; therefore, *Agency*2 can provide the provenance of R2 to *Agency*1, but protect it by using access control or redaction policies.

3.4 Policy Specification and Enforcement

Our architectural design supports a high level specification of policies, thus separating the business rules from a specific policy implementation.

*Agency*1 wishes to express its policies in a high-level language (e.g., XACML), and would prefer not learning RDF or any of its variations. The framework exposes a web interface layer between the users and the policy engine layer, whereby the users can specify their policies independent of the actual implementation of the policy. A suitable adapter, also known as a data translator, will translate each high-level policy specification into the appropriate RDF representation used by the appropriate policy, which protects an agency's resources.

Policies may be specified using more expressive languages than RDF, by extending RDF with a formal vocabulary, in particular a sub-language of OWL. OWL has a formal semantics that are based on description logics, a decidable fragment of first order logic. Thus, by supporting an adapter pattern, our framework is extended to handle semantic policies specified in OWL and high-level policies can be translated into a suitable sub-language of OWL using existing or custom-built translators.

4 Summary and Directions

This paper has described the design and implementation of the first of its kind AIS framework that operates in the cloud. As stated earlier, the idea is for each organization to store their data and the information sharing policies in a cloud. The information is shared according to the policies. We described a cloud-based information sharing framework that utilizes semantic web technologies. Our framework consists of a policy engine that reasons about the policies for information sharing purposes and a secure data engine that stores and queries data in the cloud. We also described the operation of our system with example policies.

Our framework is flexible so that additional data sources and cloud can be added. Furthermore, by using RDF for a policy engine, we can add more sophisticated policies for information sharing. This is one of the major strengths of our system. Future directions include specifying and reasoning about more sophisticated policies as well as testing our system in a real-world environment.

References

1. Thuraisingham, B., Khadilkar, V., Rachapalli, J., Cadenhead, T., Kantarcioglu, M., Hamlen, K., Khan, L., Husain, F.: Cloud-Centric Assured Information Sharing. In: Chau, M., Wang, G.A., Yue, W.T., Chen, H. (eds.) PAISI 2012. LNCS, vol. 7299, pp. 1–26. Springer, Heidelberg (2012)
2. Klyne, G., Carroll, J., McBride, B.: Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation (2004)
3. Kagal, L.: Rei. HP LabsLabs (2002), http://www.hpl.hp.com/techreports/2002/HPL-2002-270.html
4. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (January 2008)
5. Braun, U., Shinnar, A., Seltzer, M.: Securing provenance. In: Proceedings of the 3rd Conference on Hot Topics in Security, p. 4 (2008)
6. Cadenhead, T., Khadilkar, V., Kantarcioglu, M., Thuraisingham, B.: A language for Provenance Access Control. In: Proceedings of the First ACM Conference on Data and Application Security and Privacy, pp. 133–144 (2011)
7. Cadenhead, T., Khadilkar, V., Kantarcioglu, M., Thuraisingham, B.: Transforming Provenance Using Redaction. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, pp. 93–102 (2011)
8. Cadenhead, T., Khadilkar, V., Kantarcioglu, M., Thuraisingham, B.: A cloud-based RDF policy engine for assured information sharing. In: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, pp. 113–116 (2012)
9. Samarati, P., de Capitani di Vimercati, S.: Access Control: Policies, Models, and Mechanisms. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001)
10. Ehrig, H.: Fundamentals of algebraic graph transformation. Springer-Verlag New York Inc. (2006)
11. Rozenberg, G.: Handbook of graph grammars and computing by graph transformation: Foundations. World Scientific (2003)
12. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., et al.: The open provenance model core specification (v1. 1). Future Generation Computer Systems, 743–756 (2011)
13. Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3C Working Draft (2010)

Optimization of Key Distribution Protocols Based on Extractors for Noisy Channels within Active Adversaries

Victor Yakovlev[1], Valery Korzhik[1], Mihail Bakaev[1], and Guillermo Morales-Luna[2]

[1] Department of Information Security of Telecommunication Systems,
State University of Telecommunication, St. Petersburg, Russia
viyak@bk.ru, val-korzhik@yandex.ru
[2] Computer Science, Cinvestav-IPN, Mexico City, Mexico
gmorales@cs.cinvestav.mx

Abstract. We consider the information-theoretic secure key distribution problem (KDP) over noisy binary symmetric channels with public discussion and in the presence of an active adversary. There are several versions of such protocols proposed by Maurer, Wolf, Renner, Dodis, Reyzin et al. We describe two new versions of KDP for the same channel model and with the use of extractors as a mean of privacy amplification but with the goal to maximize the key rate under an optimization of the protocol parameters. There are two novelties in solution of KDP: we get the extractor's seed directly from the distributed initial strings and we prove the main results in terms of explicit estimates without the use of the uncertain symbols O, Ω, Θ. Both asymptotic and non-asymptotic cases are presented. It is shown that the extractors can be superior to conventional hashing for very large lengths of initially distributed strings.

Keywords: Active adversary, cryptography, extractors, key distribution, privacy amplification.

1 Introduction

Advances in the design of quantum computers [1] as well as in the design of super-fast multiprocessor conventional computers pose a threat to some conceptually secure cryptosystems. Hence the perfect one-time pad ciphers proposed by Shannon [2] is quite requested. But the use of perfect ciphers requires the key lengths to be proportional to the length of the messages [3]. This defect can be solved with the use of key distribution over communication channels protected from eavesdropping. There are several approaches in order to remove (or at least to control) keys eavesdropping:

- quantum cryptography [4],
- methods based on fluctuation of radio wave channels [5,6],
- the use of Wyner's wire-tap channel concept [7-9],
- key generation by hashing of random string initially distributed over noisy channels [10-20].
- key generation based on the extension of the secure sketch and fuzzy extractors technique [21-25].

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 51–64, 2012.

In the current paper, we follow the last two approaches. The most advanced results under the condition of an active adversary were obtained by Maurer and Wolf. They proposed several key distribution protocols [11-16] and made a performance comparison of asymptotic and non-asymptotic key rates for a given level of key security. A similar concept is considered in [19]. The last approach proposed at [21, 22] is now intensively developed. In [23] the key distribution protocol (KDP) has been significantly improved. The most effective solutions (in the sense of a minimization of the entropy loss and the number of rounds for interactive KDP) were presented at [24,25].

The main feature of the KDPs presented in the current paper is their exact constructability. This means that we find the optimal parameters and estimate efficiency without the use of such uncertain symbols as O, Ω, Θ and not necessarily asymptotically. Since it is a very hard problem to find exact constructive solutions for KDP in a general case, we restrict our consideration to the following conditions:

- the key distribution is performed only if the legal uses are able to get initially distributed binary i.i.d. strings over noisy channels, the eavesdropper is able to intercept these strings also over noisy channels but with different parameters
- the public discussion (in order to reconcile the keys between legal users) is executed on noiseless channel in presence of an active adversary.
- key authentication should be performed non-interactively allowing to share the secret key between more than two legal users.

The contributions of the current paper are the following:

1. We consider a modified α_{ext} - and a new β_{ext} -KDP based on extractors, different to previous ones because the extractor's seed is not transmitted over the public discussion channel but it is formed from initially distributed strings.
2. A maximization criterion is chosen as the key rate (defined as the ratio of the key length and the length of the initially distributed string) rather than the entropy loss used in [23-25]. We claim that such criterion is more practical.
3. We prove an asymptotic behavior of the key rates for new protocols allowing to compare their potential efficiency with the potential efficiency of former protocols.

The outline of this paper is the following: In Section 2 we describe the model of key distribution based on noisy wire-tap channels in the presence of an active adversary and we introduce the main criteria for the KDP efficiency. In section 3 we describe the α_{ext} -protocol, and we propose the new β_{ext}-protocol without transmission of the extractor's seed on the public discussion channel and their main features. Conclusions are at section 4. The authentication and extraction procedures appear in the Appendix.

2 Key Distribution Model and Main Criteria for Efficiency

Let us consider the model of key distribution among two legal users, *Alice* (A) and *Bob* (B), in the presence of an active adversary, *Eve* (E), assuming that initially the

legal users do not share secret keys. The *key distribution protocol* (KDP) consists of two phases: *initialization phase* and *key generation phase*.

In the KDP initialization phase, A, B, and E receive random binary i.i.d. sequences of length k: $\mathrm{X} = \{\mathrm{x}_i\}_{i=1}^{k}, Y = \{y_i\}_{i=1}^{k}, Z = \{z_i\}_{i=1}^{k} \in \{0,1\}^k$, respectively, such that for each i, the probabilities $p_m = Pr(x_i \neq y_i)$ and $p_w = \min\{\Pr(x_i \neq z_i), \Pr(y_i \neq z_i)\}$ are fixed. One method to provide legal users A, B with such sequences *X, Y* is to generate a truly random sequence $S = \{s_i\}_{i=1}^{k} \in \{0,1\}^k$ by some third party, say source **S**, and then to transmit it to the legal users A and B over noisy channels (source model [12]). We will assume that A and B receive the sequences *X, Y* over binary symmetric channels (BSC) without memory with error probabilities $\pi_A = Pr(x_i \neq s_i)$, $\pi_B = Pr(y_i \neq s_i)$, while the adversary E receives the sequence *Z* over a BSC with error probability $\pi_E = Pr(z_i \neq s_i)$. It is easy to see that if the original sequence *S* is truly random then the same property holds for the sequences *X, Y* and *Z*. Here it is natural to assume that the adversary is unable to intervene the transmission from **S** to A and B.

The key generation phase consists of an information exchange over a *public one-way channel* (POWC) from A to B with the goal to share a secret and reliable final key. After executing the string reconciliation, the legal users A and B perform a privacy amplification procedure based on extraction with a goal to get the coinciding secret keys K_A and K_B respectively. The adversary E can receive all information transmitted over the POWC and E can change or replace this information.

Let us define the following parameters of the KDP characterization:

l: the key length (the number of bits contained in the keys strings K_A and K_B),

$I(K_A, U)$: the amount of Shannon's information in possession of the adversary E about the final key K_A ,(K_B) after receiving all acceptable, for E, information U, including the sequence Z and the other messages transmitted over the POWC,

$P_e = \Pr(K_A \neq K_B)$: the probability of legal users key disagreement,

P_f : the probability of false rejection of the KDP (when A or B falsely believe that E has intervened the POWC),

P_d: the probability of deception false information provided by E during information transmission over POWC (it can result in an opportunity to fix a key between any legal user and E, although leaving this legal user on the belief that he (she) has shared a key with his (her) legal partner).

R_k: key distribution rate (the ratio of the key length l and the lengths of X^k and Y^k),

$$R_k = \frac{l}{k}. \tag{1}$$

It is reasonable to impose the following conditions on KDP:

$$l = l^{req}, \tag{2}$$

$$I(K_A; U) \leq I^{adm}, \tag{3}$$

$$P_e \leq P_e^{adm},\ P_f \leq P_f^{adm},\ P_d \leq P_d^{adm}, \tag{4}$$

where l^{req} denotes the *required key length* and the superscript *adm* stands for *admissible parameter value.* The efficiency of the KDP will be estimated by the key rate R_k given (2)-(4). We select the most efficient KDP making R_k to reach its largest value. Since the inequalities (3)-(4) may hold randomly, let us add the requirement

$$P_{risk} \le P_{risk}^{adm}, \tag{5}$$

where P_{risk}^{adm} is the probability that at least one of the inequalities (3)-(4) does not hold.

3 The New Key Distribution Protocols and Their Optimization

Two key distribution protocols in the presence of an active adversary have been proposed by Maurer and Wolf in [15]: the *UH-protocol*, in which the privacy amplification procedure was executed using hash functions and the *EX-protocol,* based on extractions. (In [21, 24, 25], the KDP using secure sketch, fuzzy extractor and robust fuzzy extractor techniques have been investigated more profoundly).

Initially we consider a modified EX-protocol called α_{ext} *–protocol*. A difference between the original and the modified protocols is determined by two factors.

1. We consider protocols under the condition $\pi_A \neq 0, \pi_B \neq 0$, $\pi_A, \pi_B < \pi_E$ or equivalently under the condition $p_m > 0$, $p_w > p_m$. This condition requires to send the check symbols from A to B in order to conciliate X^k and Y^k.
2. Instead of the authentication algorithm "*request-response*" [15], we use non-interactive algorithm based on special authentication codes (AC) [12] (see Appendix B) because it allows the users to provide authentication even when the sequences X^k and Y^k do not coincide completely and the number of legal users are greater than 2. By the same reason, the authentication algorithm and the number of substrings of the original strings X^k and Y^k are changed.

Before executing the α_{ext}-protocol, A and B divide their sequences X^k, Y^k, into $X_1^{k_1}, X_2^{k_2}$ and $Y_1^{k_1}, Y_2^{k_2}$ of respective lengths k_1, k_2. The α_{ext} –protocol proceeds as:

1. The user A forms the string $C_1^{r_1}$ of check symbols of length r_1 to the string $X_1^{k_1}$ using a (k_1+r_1, k_1)–error correcting code C_1 (agreed by the users in advance).
2. The user A generates a truly random binary sequence γ (which will be used as an extractor seed) of length u.
3. The user A forms the authenticator **w** to the message $(C_1^{r_1}, \gamma)$ using for that an AC based on error correcting $(n_0, k_0 = r_1+u, d)$- code and the sequence $X_2^{k_2}$.
4. A sends to B the pair $(C_1^{r_1}, \gamma)$ over a POWC appended with the authenticator **w**.
5. The user B verifies the authenticity of the message $(C_1^{r_1}, \gamma)$ through the known (n_0, k_0)-AC and his string $Y_2^{k_2}$ (see Appendix). If authenticity is confirmed, then B goes to the next step. Otherwise he rejects the KDP.

6. The user *B* corrects the error in string $Y_1^{k_1}$ through the check symbols string $C_1^{r_1}$. We denote by $\tilde{Y}_1^{k_1}$ the string $Y_1^{k_1}$ after error correction.
7. In order to get the keys K_A and K_B, A and B execute a privacy amplification procedure based on extractors (see Appendix A): $\mathrm{K_A} = E_{\mathrm{ext}}(X_1, \gamma)$,. $\mathrm{K_B} = E_{ext}(\tilde{Y}_1, \gamma)$

We propose also a new β_{ext} -protocol that differs from the α_{ext} –protocol in the following: After the execution of the initialization phase, both users A and B have got the strings that can in fact be used to form the seed for extractor function for privacy amplification. Hence, we do not require to send a truly random sequence γ over POWC, and the sequence's length k_2 used before for authentication of the seed can be shortened. Therefore we may expect that the length of substring X_1 is increased (if the total length of the string *X* is fixed). But such conclusion is not so apparent because we have to extract the seed as a segment of the string *X*. (It is worth to note that although γ is not uniformly distributed from the adversary's point of view this fact has no relevance for strong extractors). Let us describe the β_{ext} –protocol in detail.

In the above setup, the users A and B divide initially the strings X^k, Y^k into three disjoint parts $X_1^{k_1}, X_2^{k_2}, X_3^{k_3}$ and $Y_1^{k_1}, Y_2^{k_2}, Y_3^{k_3}$, with $k_1 + k_2 + k_3 = k$.

1. The user A forms the length r_1 string $C_1^{r_1}$ of check symbols of the string $X_1^{k_1}$ using the error correcting $(k_1 + r_1, k_1)$-code C_1, agreed in advance.
2. The user A forms the length r_2 check string $C_2^{r_2}$ to the string $X_3^{k_3}$ using the error correcting $(k_3 + r_2, k_3)$-code C_2, agreed in advance.
3. The user A forms the authenticator **w** of the message $(C_1^{r_1}, C_2^{r_2})$ using an AC and his substring $X_2^{k_2}$.
4. The user A sends to B the pair $(C_1^{r_1}, C_2^{r_2})$ over a POWC appended with **w**.
5. The user B verifies the authenticity of the message $(C_1^{r_1}; C_2^{r_2})$ using an AC and his substring $Y_2^{k_2}$. If it is confirmed then he goes to the next step. Otherwise he rejects the KDP.
6. The user B corrects errors on strings $Y_1^{k_1}$, $Y_3^{k_3}$ using the check strings $C_1^{r_1}$ and $C_2^{r_2}$. Denote by $\tilde{Y}_1^{k_1}$, $\tilde{Y}_3^{k_3}$ the strings $Y_1^{k_1}$, $Y_3^{k_3}$ after error corrections.
7. The users A and B take their substrings $X_3^{k_3}$, $\tilde{Y}_3^{k_3}$, where k_3=*u*, as the seeds γ^u in their extractors.
8. Both users A and B form the keys as $K_A = E_{ext}(X_1, X_3), K_B = E_{ext}(\tilde{Y}_1, \tilde{Y}_3)$.

Performance evaluation of KDPs and their optimization

In order to compare the performance of KDP based on extraction and KDP based on conventional hashing, (see [19]), and also to compare non-asymptotic and asymptotic results we have proven the following lemma.

Lemma 1. *If the statistical distance among the output of the extractor generating the length* l *key and an uniform distribution is at most* ε, *then the amount of the Shannon´s information concerning the key got by any adversary is bounded by*

$$I(K^l; Z^k / \Gamma^u) \le 2l\sqrt{\varepsilon}$$

(It can be easily proved with the use of Markov inequality).

It follows from lemma 1 that a requirement, regarding the amount of Shannon's information on the key leaking to an adversary, of the form $I(K^l; Z^k / \Gamma^u) \le I^{adm}$ holds if $2l\sqrt{\varepsilon} = I^{\text{adm}}$. This fact results in the following requirement to the extractor's statistical distance:

$$\varepsilon = \left(\frac{I^{\text{adm}}}{2l} \right)^2 . \tag{6}$$

Let us prove a theorem for the both α_{ext} *and* β_{ext}-protocols. We will assume that for the α_{ext}, and β_{ext}-protocols a modified Trevisan extraction is used [26,27], where the number of random bits u is determined by equation (A6), see Appendix A.

Theorem 1. *Let us assume that the users A, B and the adversary E have binary strings* X^k, Y^k *and* Z^k, *respectively after execution of the initialization phase over the wire-tape channel,* $p_m = p(x_i \ne y_i)$, $p_w = \min(p(x_i \ne z_i); p(y_i \ne z_i))$, $p_m \ge 0$, $p_w > p_m$. *Then A and B are able to form a common key of length l satisfying the requirements* (3)-(4) *after the execution of the* α_{ext} *- or* β_{ext}*-protocols if the lengths* k_1, k_2 *of parts on which were divided the substrings* X^k, Y^k *for* α_{ext} *-protocol or the parts of lengths* k_1, k_2, k_3 *on which were divided the substrings* X^k, Y^k *for* β_{ext}*-protocol satisfy the relations listed below:*

$$k_1 = \frac{-\log P_e^{adm}}{E(R_{c_1})} \quad \text{for } \alpha_{ext} \text{ and } \beta_{ext} - \text{protocols}, \tag{7}$$

$$k_1 \cdot H_\infty = lc + r_1 - \log P_{\text{risk}}^{\text{adm}} + u + 3\log\left(\frac{l}{(I^{adm}/2l)^2}\right) + 3 \text{ for } \alpha_{ext} \text{ and } \beta_{ext} - \text{protocols} \tag{8}$$

where $E(R_{c1})$ *is the modified Gallager's function for BSC [28],* r_1 *is the number of check symbols of the* $(k_1 + r_1, k_1)$*-error correcting code* C_1,

$$u = \left\lceil \frac{\left\lceil \log \frac{k_1}{(I^{adm}/2l)^2} \right\rceil}{\ln c} \right\rceil \cdot \left\lceil \log \frac{k_1}{(I^{adm}/2l)^2} \right\rceil, \tag{9}$$

is the number of extractor random symbols, c is a parameter under optimization, $H_\infty = -\log \max(p_w, 1 - p_w)$ *is the min entropy,*

$$k_2(1-g\left(\frac{2d}{k_2}\right)) = 2k_0, \tag{10}$$

$$\sum_{i=\Delta_w+1}^{k_2}\binom{k_2}{i} p_m^i (1-p_m)^{k_2-i} = P_f^{adm}, \tag{11}$$

$$\sum_{i=0}^{\Delta_w}\binom{d}{i} p_w^i (1-p_w)^{d-i} \cdot \sum_{j=0}^{\Delta_w - i}\binom{k_2-d}{j} p_m^j (1-p_m)^{k_2-d-j} = P_d^{adm}, \tag{12}$$

where

$$k_0 = \begin{cases} u+r_1 & \text{for the } \alpha_{ext}-\text{protocol}, \\ r_1+r_2 & \text{for the } \beta_{ext}-\text{protocol} \end{cases} \tag{13}$$

and r_2 being the number of check symbols of the (k_3+r_2, k_3)-error correcting code C_2 that is found similarly as in eq's (7), d is the minimal Hamming distance of the code C_2,

$$k_3 = \begin{cases} 0 & \text{for the } \alpha_{ext}-\text{protocol}, \\ u & \text{for the } \beta_{ext}-\text{protocol} \end{cases} \tag{14}$$

The key rate is then determined as follows:

$$R_{\alpha_{ext}} = \max_c \frac{l}{k_1+k_2}, \quad R_{\beta_{ext}} = \max_c \frac{l}{k_1+u+k_2}. \tag{15}$$

Proof. Let the bounds of the KDP parameters meet exactly all requirements (3), (4).

Initially we prove the relation (7).

Let C_1 be a binary linear error correcting code and let $C_1^{r_1}$ be a string consisting of r_1 check symbols, then $R_{c1} = \frac{k_1}{k_1+r_1}$ is the code rate. It has been proved in [28] that if the information symbols are transmitted on the BSC with the error probability p_m, whereas the check symbols are transmitted on the noiseless channel, then the average error probability of decoding on the ensemble of all (k_1+r_1, r_1)-codes meets the following modified Gallager's bound

$$P_{e1} \le 2^{-k_1 E(R_{c1})}, \tag{16}$$

where $E(R_{c_1})$ is Gallager's function for BSC with the error probability p_m

$$E(R_{c1}) = \max_{\rho\in(0,1)}\left[E_0(\rho) - \frac{\rho(2R_{c1}-1)}{R_{c1}}\right] \tag{17}$$

$$E_0(\rho) = \rho - (1+\rho)\log_2\left(p_m^{\frac{1}{1+\rho}} + (1-p_m)^{\frac{1}{1+\rho}}\right) \tag{18}$$

Letting $P_{e1} = P_e^{adm} = 2^{-k_1 E(R_{c1})}$ in (16), we get (7),

In order to prove (8) we note the following.

Under the condition that the adversary gets the sequence Z^{k_1} over a BSC with error probability p_w the conditional minimal entropy is

$$H_\infty(X^{k_1} \mid Z^{k_1}) = k_1 H_\infty(X \mid Z) = -k_1 \log max(p_w, (1-p_w)) = k_1 H_\infty .$$

Since the adversary receives also the check block $C_1^{r_1}$, in line with corollary 2 in [15] the following inequality results:

$$H_\infty(X_1^{k_1} \mid Z_1^{k_1}, C_1^{r_1}) \geq k_1 H_\infty - r_1 - s , \tag{19}$$

that does not hold with the probability $P_{risc} \leq 2^{-s}$.

Using proposition 10 in [27] (see also (A5) in Appendix) and (19) we may write $\varepsilon \leq 2^{\tau/3}$, where $\tau = lc + 3\log l - \mathrm{k_1 H_\infty} + r_1 + s + u + 3$. Let us assume that I^{adm} is chosen in such a way, that $2^{\tau/3} = \log(\frac{I^{adm}}{2l})^2$, resulting thus the condition (6). Hence,

$$lc + 3\log l - \mathrm{k_1 H_\infty} + r_1 + s + u + 3 = 3\log\left(\frac{I^{adm}}{2l}\right)^2 . \tag{20}$$

Assuming $P_{risk} = P_{risk}^{adm} = 2^{-s}$, (8) holds eventually from (20).

The value *u* in (9) is the number of the extractor random symbols. In order to find it, we can use the results of [27] and substitute (A4) into (A6) and then substitute ε from (6).

A solution of the equation system (7)-(8) allows to find the parameters k_1, r_1, given a fixed *c*. It will be shown in the sequel that the key rate can be maximized by a proper selection of the parameter *c*.

In order to find k_2 let us assume that the probabilities P_f and P_d have equal values $P_f = P_f^{adm}$, $P_d = P_d^{adm}$. It is easy to see that

$$P_f^{adm} = \sum_{i=\Delta_w+1}^{2n_0} \binom{2n_0}{i} p_m^i (1-p_m)^{2n_0-i}, \tag{21}$$

$$P_d^{adm} = \sum_{i=0}^{\Delta_w} \binom{d}{i} p_w^i (1-p_w)^{d-i} \cdot \sum_{j=0}^{\Delta_w - i} \binom{2n_0 - d}{j} p_m^j (1-p_m)^{2n_0-d-j}, \tag{22}$$

where n_0, k_0, d are the parameters of error correcting codes used in the AC [29].

Recall that for the AC (see Appendix B) we have $k_2 = 2n_0$ where n_0 is the length of the error correcting (n_0, k_0)-code with minimum distance *d*. For the α_{ext}-protocol $k_0 = r_1 + u$, while for the β_{ext}-protocol $k_0 = r_1 + r_2$, where r_2 is the number of check symbols in the $(k_3 + r_2, k_3)$-code C_2. This gives relation (13) for the parameter k_0. Using the Varshamov-Gilbert inequality connecting n_0, k_0, d and taking into account that $k_2 = 2n_0$ we get

$$k_2(1 - g\left(\frac{2d}{k_2}\right)) = 2k_0 \tag{23}$$

By solving the equation system (21)-(23), which is equivalently to the system (10)-(12), we find the parameters k_2, d.

In line with the above protocols, we have that for the α_{ext}-protocol, $k_3 = 0$ and for the β_{ext}–protocol $k_3 = u$. This fact proves (14).

The value r_2 is calculated by (17)-(18), for $(k_3 + r_2, k_3)$-code C_2 in which it is necessary to let $k_1 = k_3 = u$, $r_1 = r_2$, $p = p_m$, $P_{e2} = P_e^{adm}$.

Then relation (15) is apparent from the protocols description. The KDP rates can be maximized by a selection of extractor parameter c. □

Remark. *If the solution of system (7)-(14) is not unique then it is reasonable to select any of them by maximizing the key rate.*

Three corollaries follow easily from theorem 1.

Corollary 1. *If the key length* l *is given and the rate* R_c *of the error correction code satisfies inequality* $R_c \geq 2/3$, *then* $R_{\beta_{ext}} \geq R_{\alpha_{ext}}$.

Corollary 2. *As the key length* $l \to \infty$, *then the following relation holds*

$$R_{\alpha_{ext}} = R_{\beta_{ext}} = \frac{H_\infty(p_w) - g(p_m)}{1 + 2g(p_m)} \tag{24}$$

The following corollary allows to compare the key rates of the α_{ext} -, β_{ext} –protocols with the key rates of the α -, β- protocols, obtained in [19].

Corollary 3. *If* $p_m = 0$, *then* $R_{\alpha_{ext}} \geq R_\alpha$ *and* $R_{\beta_{ext}} \geq R_\beta$.

In Fig. 1 we plot the key rates R_k versus the length l for both α_{ext} -, and β_{ext} -protocols with typical parameters p_m = 0.01 and 0.001, p_w = 0.2, and the requirements I^{adm} = 10^{-30}, $P_e^{adm} = P_d^{adm} = P_f^{adm} = P_{risk}^{adm} = 10^{-5}$.

The optimization of c has been performed for every value of l. For comparison purposes the dependences $R_\alpha(l)$ and $R_\beta(l)$, obtained in [19] are shown also in the figure (see α- and β-protocols).

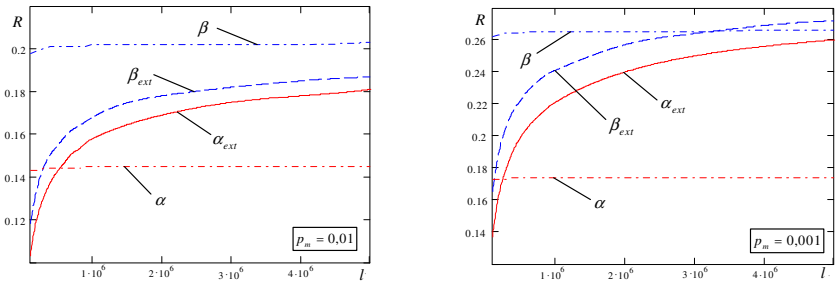

Fig. 1. The key rates versus key lengths for different requirements imposed to KDPs and four types of KDP (α_{ext}, β_{ext}. and α, β)

The following claim can be drawn after an examination of the obtained dependences.

The protocols using extractors have greater key rate than the α- and β-protocols under sufficiently large l and small p_m. Then more specifically the α_{ext}- is better than the α- protocol when $l > 5 \cdot 10^5$ with p_m=0.01 and the β_{ext}- protocol is better than the β- protocol when $l > 3{,}5 \cdot 10^6$ and p_m=0.001 (under the requirements stated in our investigations). The key length for which the α_{ext}- and β_{ext}-protocols are superior to the α- and β-protocols essentially depends on the error probabilities in the communication channels. The β_{ext}- protocol is superior to the α_{ext} -protocol, although these protocols have the same asymptotic key rate. Also, protocols with extractors are always superior to the protocols with hashing, as $p_m = 0$.

Thus we may claim that the new β-, β_{ext} -KDP in which the number of hash function or extractor's seed are not transmitted over public channel but they are formed from initially distributed strings have larger key distribution rates under some conditions than the key rates for α-, α_{ext} -protocols which are in turn modified protocols initially proposed in [15].

4 Conclusion

In the current paper, an investigation of key distribution protocols based on noisy channels started in [19] has been continued in such a way that extractors were used instead of hash-functions in the privacy amplification procedure. The main goal was to prove extractor-based protocols efficiency on the criterion of key rate maximization. The relations are not necessary asymptotic and they are constructive because in contrast to other papers they do not include uncertain symbols.

It has been proposed new β_{ext}-protocol that differs from those known before [15] because the extractor's seed is not transmitted over the POWC and, instead, it is generated from random sequences obtained by legal user after the execution of the initialization phase. We proved that the use of extractors in the α_{ext} - and β_{ext}-protocols increases the rate, in comparison with hashing-based protocols only for very large key length l (typically $l = 10^5 - 10^6$) and for some specified values of the error probabilities both in the main and in the wire-tap channels.

We get also asymptotic estimates for the key rates of proposed protocols allowing to compare a potential efficiency of the considered early protocols.

References

1. Knill, E.: Bulding Quantum Computers, 2007 IEEE Int. Symp. on Informational Theory. IEEE Information Theory Society Newsletter 58(4), 32–35 (2008)
2. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal 28(4), 656–715 (1949)
3. Hellman, M.E.: An extension of the Shannon theory approach to cryptography. IEEE Transactions on Information Theory 23(2), 289–294 (1977)
4. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: Proc. Int. Conf. on Computers, Systems & Signal Processing (1984)

5. Aono, T., Higuchi, K., Ohira, T., Komiyama, B., Sasaoka, H.: Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. IEEE Transactions on Antennas and Propagation 53(11), 3776–3784 (2005)
6. Yakovlev, V., Korzhik, V., Kovajkin, Y., Morales-Luna, G.: Secret Key Agreement Over Multipath Channels Exploiting a Variable-Directional Antenna. Int. Jour. Adv. Computer Science & Applications 3(1), 172–178 (2012)
7. Wyner, A.: Wire-tap channel concept. Bell System Technical Journal 54, 1355–1387 (1975)
8. Korjik, V., Yakovlev, V.: Non-asymptotic estimates for efficiency of code jamming in a wire-tap channel. Problems of Information Transmission 17, 223–22 (1981)
9. Korjik, V., Yakovlev, V.: Capacity of communication channel with inner random coding. Problems of Information Transmission 28, 317–325 (1992)
10. Bennett, C.H., Brassard, G., Crepeau, C., Maurer, U.M.: Generalized privacy amplification. IEEE Transactions on Information Theory 41(6), 1915–1923 (1995)
11. Maurer, U.M.: Secret key agreement by public discussion from common information. IEEE Transactions on Information Theory 39(3), 733–742 (1993)
12. Maurer, U.M.: Information-Theoretically Secure Secret-Key Agreement by NOT Authenticated Public Discussion. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 209–225. Springer, Heidelberg (1997)
13. Maurer, U.M.: Protocols for Secret Key Agreement by Public Discussion Based on Common Information. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 461–470. Springer, Heidelberg (1993)
14. Maurer, U.M., Wolf, S.: Privacy Amplification Secure against Active Adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 307–321. Springer, Heidelberg (1997)
15. Maurer, U.M., Wolf, S.: Secret-key agreement over unauthenticated public channels iii: Privacy amplification. IEEE Trans. Information Theory 49(4), 839–851 (2003)
16. Maurer, U.M., Wolf, S.: Towards Characterizing when Information-Theoretic Secret Key Agreement is Possible. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 196–209. Springer, Heidelberg (1996)
17. Korzhik, V., Yakovlev, V., Sinuk, A.: Achieveability of the Key-Capacity in a Scenario of Key Sharing by Public Discussion and in the Presence of Passive Eavesdropper. In: Gorodetsky, V., Popyack, L.J., Skormin, V.A. (eds.) MMM-ACNS 2003. LNCS, vol. 2776, pp. 308–315. Springer, Heidelberg (2003)
18. Korzhik, V., Yakovlev, V., Sinuk, A.: Key Distribution Protocol Based on Noisy Channel and Error Detecting Codes. In: Gorodetski, V.I., Skormin, V.A., Popyack, L.J. (eds.) MMM-ACNS 2001. LNCS, vol. 2052, pp. 242–250. Springer, Heidelberg (2001)
19. Yakovlev, V., Korzhik, V., Morales-Luna, G.: Key Distribution Protocols Based on Noisy Channel in Presence of Active Adversary: Conventional and New Versions with Parameter Optimization. IEEE Transaction on Information Theory 54(6), 2535–2549 (2008)
20. Yakovlev, V., Korzhik, V., Morales-Luna, G.: Non-asymptotic Performance Evalua-tion of Key Distribution Protocols Based on Noisy Channels in Presence of Active Adversary. In: Proc. X Spanish Meet. Cryptology and Information Security, Salamanca, pp. 63–68 (2008)
21. Renner, R., Wolf, S.: Unconditional Authenticity and Privacy from an Arbitrarily Weak Secret. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 78–95. Springer, Heidelberg (2003)
22. Renner, R., Wolf, S.: The Exact Price for Unconditionally Secure Asymmetric Cryptography. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 109–125. Springer, Heidelberg (2004)

23. Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 232–250. Springer, Heidelberg (2006)
24. Kanukurthi, B., Reyzin, L.: Key Agreement from Close Secrets over Unsecured Channels. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 206–223. Springer, Heidelberg (2009)
25. Chandran, N., Kanukurthi, B., Ostrovsky, R., Reyzin, L.: Privacy amplifcation with asymptotically optimal entropy loss. Cryptology ePrint Archive (2010), http://eprint.iacr.org/2010/
26. Trevisan, L.: Construction of extractors using pseudo-random generator. In: Proceedings of the 31 Annual ACM Symposium on Theory of Computing, Atlanta, pp. 141–148 (1999)
27. Raz, R., Reingold, O., Vadhan, S.P.: Extracting all the randomness and reducing the error in trevisan's extractors. J. Comput. Syst. Sci. 65(1), 97–128 (2002)
28. Korjik, V., Morales-Luna, G., Balakirsky, V.: Privacy Amplification Theorem for Noisy Main Channel. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 18–26. Springer, Heidelberg (2001)
29. Korjik, V., Yakovlev, V., Chesnokov, R., Morales-Luna, G.: Performance Evaluation of Keyless Authentication Based on Noisy Channel. In: International Conference of Mathematical Methods, Models and Architectures for Computer Network Security. CCIS, vol. 1, pp. 115–126 (2007)

Appendix

A. ***Extractors***

The literature on the subject of extractors is quite extensive. But we quote only some results which are very necessary for an understanding of the main part of the paper.

Let us recall the notions connected with *extractor and strong extractor*.

Two probability distributions *P*, *Q*, defined on the same set *X*, are called ε-close if their statistical difference $dif(P,Q)=\frac{1}{2}\sum_{x\in X}\left|P_X(x)-Q_X(x)\right|$ does not exceed ε.

A map $E:\{0,1\}^k\times\{0,1\}^u\to\{0,1\}^l$ is an (η,ε)*-extractor* if for any probability distribution variable X on $\{0,1\}^k$ such that min entropy $H_\infty(X)\ge\eta$ and for uniformly distributed variable Γ on $\{0,1\}^u$, the statistical difference probability distribution of extractor output $E(X,\Gamma)$ to an uniform distribution Γ on $\{0,1\}^l$ is at most ε. The sequence Γ can be seen as a seed of the extractor.

The extractor $E(X, \Gamma)$ has parameters (k, η, u, l, ε), where k is the length of input random sequence, η is the bound of min entropy ($H_\infty(X^k)\ge\eta$) on the set of input sequences, u is the length of the seed γ, l is the length of the output sequence, ε is the statistical distance between the output probability distribution and the uniform distribution on the output set.

A mapping $E:\{0,1\}^k\times\{0,1\}^u\to\{0,1\}^l$ is called a *strong extractor* $E(X,\Gamma)$ if for any probability distribution random variable X on the set $\{0,1\}^k$ having minimal entropy $H_\infty(X)\ge\eta$ and for uniformly distributed random variable Γ on the set

$\{0,1\}^u$ the probability distribution of the concatenated variables ($\Gamma \circ E(X,\Gamma)$) is close to an uniform distribution on $\{0,1\}^{l+u}$. More specifically

$$dis(\Gamma^u \circ Ext(X,\Gamma), V^{u+l}) \le \varepsilon . \quad \text{(A1)}$$

This means that the strong extractor provides the closeness of probability distribution for the concatenation of the output extractor sequence and the seed sequence to an uniform distribution.

In the current paper we consider only extractors based on the construction proposed by Trevisan and improved later [26], [27].

We are not going to use the estimates based on the O-operator and therefore let us find a more accurate estimate for the length of the seed. In order to design the Trevisan's extractor it is necessary to realize three components:

1. The linear error $(\tilde{n},k)$-code W with minimal code distance d_w, where $\tilde{n} = 2^v$, v- integer. It is proposed to take this code as a concatenation of the Reed-Solomon and the Adamar codes.
2. Combinatorial block design scheme. (*Balance incomplete block design*, BIBD). This is a family of sets $S = \{S_1, S_2, \dots S_l\}$ holding the following properties:

$$S_i \subseteq [1,2,\dots u], \ |S_i| = v, \ \sum_{j<i} 2^{|S_i \cap S_j|} \le c(l-1) \quad \text{(A2)}$$

where c is some constant ($c>1$). This means that a family consists of l sets or blocks, each consisting of v elements taken from the set of integer $\{1,2,\dots,u\}$, while the number of elements contained simultaneously in any pair of blocks satisfies of last relation in (A2). Such construction is defined as a (v, c)-*weak scheme*.

3. Boolean function f with v arguments $a_1, a_2, \dots a_v$, such that $f(a_1, a_2, \dots a_v) = \overline{w}$, where $\overline{w}$ is a codeword of the $(\tilde{n},k)-$code W. The W-code length $\tilde{n}$ is chosen in [27, p.106] according to the condition,

$$\log\tilde{n} = O(\log k/\varepsilon) \quad \text{(A3)}$$

Since w is the output of Boolean function with v arguments, $\tilde{n}$ should be equal to 2^v. Obviously the condition (A3) holds if

$$v = \left\lceil \log \frac{k}{\varepsilon} \right\rceil, \quad \text{(A4)}$$

where $\lceil x \rceil$ is the "ceiling" of x (the least integer greater or equal than x).

The characterization of strong extractor is determined by the following statements:

(Proposition 10 in [27]). *If* $\mathcal{S} = \{S_1,\dots,S_l\}$ *(with* $S_i \subset \gamma$*) is a weak (v,c)-design for*

$$c = (H_\infty(X^k) - 3\log(l/\varepsilon) - u - 3)/l, \quad \text{(A5)}$$

then $E : \{0,1\}^k \times \{0,1\}^u \to \{0,1\}^l$ *is a strong* $(H_\infty(X^k),\varepsilon)$-extractor.

(Lemma 15 in [27]). *For every v,l∈* $\mathbb{N}$ *and c>1, there exists a weak (v,c)-design* $\mathcal{S}$ $=\{S_1,\ldots,S_l\}$ ($S_i \subset \gamma$) *with*

$$u = \left\lceil \frac{v}{lnc} \right\rceil \cdot v \cdot \tag{A6}$$

Moreover, such a family can be found in polynomial time poly(l,u).

B. Authentication Based on Noisy Channels

In [12] a special type of codes (AC) has been proposed for an authentication on noisy channels.

They can be characterized by two probabilities [29]:

P_f – the probability of false removal of the message although adversary E does not intervene at all;

P_d – the probability of the deception of false message, i.e. the probability of the event that E is a forged message and this fact was not detected by B.

It has been proved [12] that P_f P_d, do not depend on ordinary minimum code distance of the code V but on the so called minimum asymmetric semidistance d_{01} that is determined by the minimal number of differences between 0 and 1 symbols in any pair of distinct code words of V.

Theorem [29]. *Let V be an* (n_a, k_a)*-AC with constant Hamming weight τ for all non-zero codewords and with asymmetric semidistance* d_{01}*. Then the probabilities* P_f, P_d *for authentication procedure on noisy wire-tap channel with parameters* p_m, p_w *can be upper bounded as follows:*

$$P_f \le \sum_{i=\Delta_w+1}^{\tau} \binom{\tau}{i} p_m^i (1-p_m)^{\tau-i} \tag{A7}$$

,

$$P_d \le \sum_{i=0}^{\Delta_w} \binom{d_{01}}{i} p_w^i (1-p_w)^{d_{01}-i} \cdot \sum_{j=0}^{\Delta_w - i} \binom{\tau - d_{01}}{j} p_m^j (1-p_m)^{\tau - d_{01} - j} \tag{A8}$$

.

It is a very hard problem to find d_{01} for any linear code. But there exists a simple method proposed in [12] to design the code V with known d_{01}, given the linear (n_0, k_0)-code $\tilde{V}$ with known ordinary minimum code distance d.

Namely, let us substitute the symbol 1 with the symbol pair 10 and the symbol 0 with 01 in V'.Then evidently the parameters of the code V are:

$$n_a = 2n_0, k_a = k_0, d_{01} = d, \tau = n_0. \tag{A9}$$

We have proved in [29] that the length of the authenticator approaches zero as the block length tends to infinity.

A Vulnerability in the UMTS and LTE Authentication and Key Agreement Protocols

Joe-Kai Tsay and Stig F. Mjølsnes

Department of Telematics
Norwegian University of Sciences and Technology, NTNU
{joe.k.tsay,sfm}@item.ntnu.no

Abstract. We report on a deficiency in the specifications of the Authentication and Key Agreement (AKA) protocols of the Universal Mobile Telecommunications System (UMTS) and Long-Term Evolution (LTE) as well as the specification of the GSM Subscriber Identity Authentication protocol, which are all maintained by the 3rd Generation Partnership Program (3GPP), an international consortium of telecommunications standards bodies. The flaw, although found using the computational prover CryptoVerif, is of symbolic nature and could be exploited by both an outside and an inside attacker in order to violate entity authentication properties. An inside attacker may impersonate an honest user during a run of the protocol and apply the session key to use subsequent wireless services on behalf of the honest user.

Keywords: Applied Cryptography, Vulnerability Assessment, Security Protocols, Authentication, Mobile Network Security, LTE, UMTS.

1 Introduction

The Global System for Mobile communication (GSM) and UMTS mobile networks are a worldwide success with now about 6 billion supscriptions [17], and still growing. New mobile systems are rolled out, including the 3GPP recent developments named 'Long Term Evolution' (LTE) and 'System Architecture Evolution' (SAE), which have become a forerunner for the fourth generation (4G) generation mobile communication system. The new system is called 'Evolved Packet System (EPS), emphasizing the all-IP packet switching design throughout the system onto the user's mobile terminal. [1] As more and more people take advantage of the accelerated internet access through their mobile phones, the recent international concern about securing the cyberspace and critical infrastructures certainly must include mobile networks. There is a multitude of security issues in such large networked systems. Here we will focus on the mobile terminal access security by means of an authentication and key agreement protocol. Weaknesses in this protocol may not only lead to revenue loss to mobile operators but might also facilitate cyber crime.

[1] Although EPS is the proper technical term for this new 3GPP mobile system generation of SAE/LTE, we will use the most well-known name LTE.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 65–76, 2012.

The LTE AKA protocol is based on the Universal Mobile Telecommunications System (UMTS) AKA protocol, which is widely used today for third generation (3G) wireless networks, and which itself is the successor of the GSM Subscriber Identity Authentication (SIA) protocol. With the persistent spread of these mobile network systems, these authentication protocols have become some of the most widely used security protocols today. While there exist formal analyses of UMTS AKA in the *Symbolic Model* of security (also called the *Dolev-Yao* model and inspired by [15]), this is in fact the first analysis of LTE AKA to date.

We report on an intermediate result of an ongoing analysis [19] of UMTS AKA and LTE AKA with the tool CryptoVerif [14] that can prove the security of protocols directly in the computational model. We discover a previously undetected flaw in the specifications of both UMTS AKA and LTE AKA. We note that the specifications of the GSM SIA protocol [9,8] suffer, strictly speaking, from the same vulnerability (cf. Section 3.3). The vulnerability can be exploited by both outside and inside attackers in order to break authentication of a user to a serving network. Furthermore, inside attackers may impersonate an honest user and use wireless services on his behalf without the user being present on the network at that time. We reported the vulnerability to the 3GPP where the issue is currently under investigation. We have not tested current implementations for susceptibility to these attacks (cf. Section 3.1). We propose a simple correction to UMTS/LTE AKA and are working on CryptoVerif proofs of correspondence (*i.e.* authentication) and secrecy properties for the session key.

Related Work. Annex B of the 3GPP technical report in [1] documents a formal analysis of the UMTS AKA protocol using a BAN logic variant. The analysis verifies authentication and secrecy properties. The flaw that we present here is not detected in [1] because strong assumptions (the *prerequisites on SN's side*) are used which already eliminate the weakness in the protocol. The GSM SIA protocol does not provide authentication of the access network to the user and the interoperability of the GSM and UMTS systems perpetuates this attack possibility, reported in [18]. Our analysis is not directed to the problems of interoperability between LTE/UMTS/GSM. A redirection attack on the UMTS AKA is reported in [20], which exploits the observation that the user is not able to authenticate the identity of the *serving* network because this is not included in the authentication vector provided by the home network. The new LTE AKA specification is designed to fix this weakness. A recent paper focuses on the privacy properties of the UMTS AKA protocol [10]. They use the tool ProVerif [12] for a formal analysis, and the paper describes an attack that enables the adversary to track a user. This is done by exploiting different error messages that are returned by UMTS AKA. The analysis models the UMTS AKA as a simplified two-party protocol between a user and the core network. However, by reducing UMTS AKA to a two-party protocol, the weakness uncovered in the present work is concealed.

Structure of This Work. In Section 2, we will give an overview of the Mobile Network architecture and give a description of the UMTS AKA and LTE AKA protocols. In Section 3, we describe the flaw we found in the specifications of UMTS and LTE AKA and its consequences, where we discuss the relevance of the flaw for GSM SIA in Section 3.3. We conclude in Section 4.

2 UMTS and LTE Authentication and Key Agreement

2.1 Overview of the Mobile Network Architecture

For both UMTS and LTE the basic network architectures are very similar. In comparison to UMTS, the network elements used for LTE are upgraded and mostly renamed. However, they fulfill the analogous tasks in both cases. In order to avoid unnecessary confusion over terminology, we give a unified description of the network architectures of UMTS and LTE at the level of detail necessary for understanding our analysis and the vulnerability presented below. Basically, the mobile network architecture comprises three parts, that is, the user's mobile equipment U, the Radio Access Network (RAN), and the Core Network (CN). The user equipment consists of the mobile equipment and a tamper-resistant chip card, the *Universal Subscriber Identity Module* (USIM). The USIM is issued by a mobile operator to a subscriber and contains the International Mobile Subscriber Identity (IMSI), the permanent key of the subscription shared between subscriber and operator, and the cryptographic algorithms for the authentication protocol. In the following, we will use the terms *user*, *subscriber* and *user equipment* interchangeably. Each mobile operator runs an *Authentication Center* (AuC) server within its core network that contains the security related information of all the subscribers of the operator and generates temporary security credentials to be used by a user and a core network to establish authentication guarantees and set up session keys. The core network is divided into a *serving network* S and a *home network* H, where the latter contains and maintains the AuC and the serving network is responsible for the communication to the user equipment through the radio access network.

The serving network and the home network do not necessarily belong to the same security domain, *i.e.* they may be controlled by different mobile operators. A subscriber U_1 of a mobile operator OP_1 with home network H_1 may *roam* into the domain of mobile operator OP_2's radio access network maintained by serving network S_2. If OP_1 has a roaming agreement with OP_2, then U_1 will be able to access the mobile network through S_2's radio access network. In this case, the connections between S_2 and H_1 are called *inter-domain connections*.

2.2 The UMTS & LTE AKA Protocols

Figure 1 shows the message sequence diagram description of the authentication and key agreement protocol in a unified way for UMTS and LTE on a similar level of detail as depicted in [3,7]. The protocol is executed between user U, visited serving network S and U's home network H. U and H share the long-term

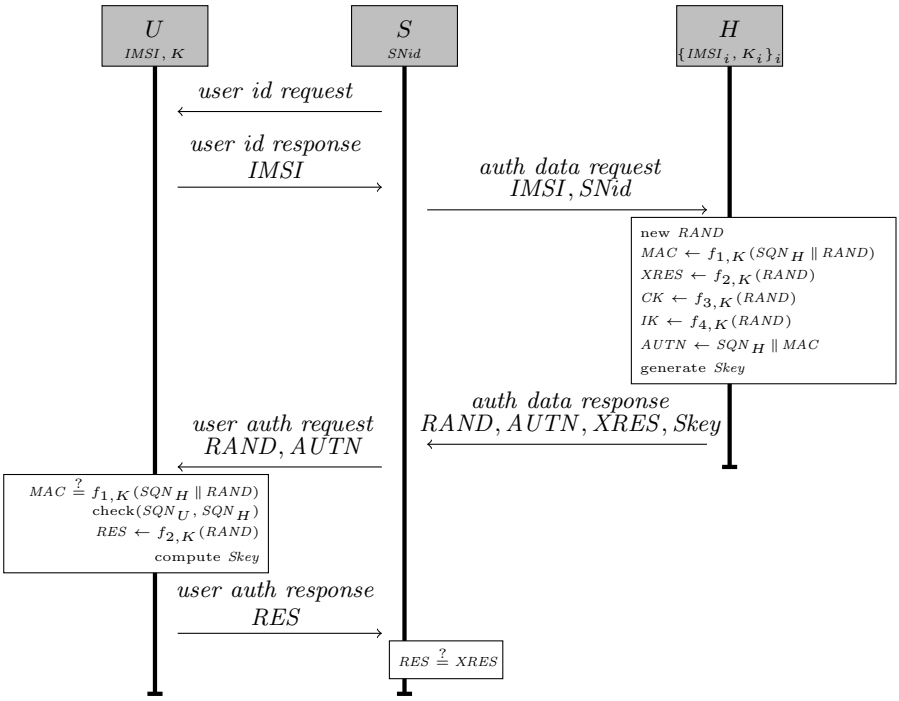

Fig. 1. The UMTS/LTE Authentication and Key Agreement Protocol. The session key in UMTS is $Skey \leftarrow CK \,\|\, IK$, and in LTE it is $Skey := K_{ASME} \leftarrow KDF(SQN_H \,\|\, CK \,\|\, IK \,\|\, SNid)$.

key K_0 and a set of algorithms $f_1, \ldots, f_4$ and, in the case of LTE, also a key derivation function KDF. The functions f_1, f_2 are so called *message authentication functions*, and f_3, f_4 are so called *key generating functions*[2]. Moreover, U maintains a counter SQN_U and H a counter SQN_H for U.

A protocol run starts with S sending a *user id request* and U responding with its *IMSI*[3]. Next follows the *authentication data transfer*, in which S sends an *authentication data request* to H, that consists of U's *IMSI* and S's identifier *SNid*, and H answers with an *authentication data response*. H chooses a fresh nonce $RAND$ and computes, with the key K and its sequence number SQN_H, the so-called *message authentication code* MAC, the *expected response* $XRES$, the *cipher key* CK, the *integrity key* IK, and the *authentication token* $AUTN$ as depicted in Figure 1, where $\|$ denotes concatenation. The main difference between the UMTS AKA and LTE AKA is the *session key* $Skey$. In LTE AKA, the session key is computed over the identifier of S (cf. caption of

[2] We choose to do without the *anonymity key*, *i.e.* $f_5 \equiv 0$, which is an option in the specifications. We also omit the AMF constant.

[3] In fact, U may alternatively respond with a temporary mobile subscriber identity (TMSI), which reduces but does not fully avoid the use of the *IMSI*.

Figure 1). There is also the option that H sends S multiple *authentication vectors* $(RAND_i, AUTN_i, XRES_i, Skey_i)$ for $i = 1, \ldots, n$ at once in order to reduce the traffic between S and H but we will not focus on the use of this option.

In the *user authentication request*, S forwards only $RAND$ and $AUTN$ to U. From the received $RAND$, $AUTN$, the user U extracts SQN_H, computes the *expected message authentication code* $XMAC$ and compares it to MAC contained in $AUTN$. If they are equal then U performs a check on the sequence numbers SQN_H and SQN_U[4]. If either of this two checks fail, then U sends some error messages to S (in fact, the error messages may be different, therefore allowing the linkability attack of [10]). Otherwise U computes the *response* RES and sends it to S. Finally, S compares the response received from U with the expected response received from H; if they are equal then the UMTS/LTE AKA run was successfully completed.

Intuitively, the UMTS/LTE AKA establishes the session key *Skey* between U and S, therefore, *Skey* must satisfy some secrecy property. Furthermore, the protocol aims to authenticate U to S. Both properties require S to trust H to provide a correct authentication data response. The sequence numbers allow to detect possible replays of authentication tokens. The UMTS/LTE AKA protocol, as depicted in Figure 1, does not offer authentication of S to U. This known weakness has been described in [20]. User U may at most know that H generated the received nonce and authentication token for some service network.

Following the UMTS/LTE AKA, serving network S and user U need to negotiate the cryptographic algorithms (*security mode*) used to protect subsequent wireless communication between S and U. Note that these algorithms are, in particular for inter-domain connections, not pre-determined. The messages of this negotiation are protected by (keys derived from) *Skey*. This is especially relevant for the case of LTE, where *Skey* is generated over S's identifier *SNid*. In LTE, by receiving the *NAS security mode command* directly following the user authentication response, U should be able to authenticate S, as this message constitutes a key confirmation of the session key K_{ASME}. According to [7], the NAS security mode command from S to U has following form.

$$S \longrightarrow U : eKSI, \textit{UE sec capabilities}, \textit{ciphering algo}, \textit{integrity algo}, \textit{NAS-MAC}$$

where *NAS-MAC* is a message authentication code under a key derived from K_{ASME} over the rest of the message, which consists of non-secret components. We denote by *LTE AKA+1* the LTE AKA protocol together with this NAS security mode command message.

3 Attacking and Correcting UMTS & LTE AKA

Here we present a weakness found in the authentication protocol specifications of both UMTS and LTE AKA with the help of the tool CryptoVerif [13]. Although CryptoVerif has semantics in the computational model, the flaw in the

[4] Checking and increasing the sequence numbers can be different for UMTS and LTE.

protocols is of symbolic nature. Unlike other provers that work in the symbolic model, CryptoVerif does not output attack traces; instead we found the attack by interpreting the *last game* in a sequence of game transformations performed by CryptoVerif. It is the same flaw that is present in the specifications of both UMTS AKA and LTE AKA. Although UMTS AKA has previously been formally analyzed [10,1], none of the previous analyses have detected this flaw. How GSM SIA is affected by the flaw is discussed in Section 3.3.

3.1 Communication Security between *S* and *H*

It is obvious that the communication between S and H needs to be protected in some way, otherwise, *e.g.* if there is no confidentially protection, the exchanged session key(s) are sent in the clear. The specifications of the AKA protocols in [7] and [3] mention little about the security protection of the authentication data transfer.

However, for UMTS and LTE, the specifications [5,6] detail the protection of IP-based communication between network elements. Here a distinction is made between inter-domain communication, where standardized solutions are necessary, and *intra-domain* communication, where the communicating parties are controlled by the same mobile operator. For inter-domain connections over IP-based networks, [5,6] mandate the protection of the communication between network elements using IPsec. For intra-domain connections over IP-based networks (*i.e.* communication over Z_b *interfaces*), [5,6] state that the protection of communication is regarded as an internal issue of each domain operator. In particular, the use of IPsec for intra-domain communication between S and H is *optional*, even though the communication may involve long distance signaling.

Furthermore, in the case of UMTS, the communication between S and H can also be carried out on the *global Signaling System No. 7 network* instead of an IP-based network. The specification [4] details the protection for such communication between S and H using *Mobile Application Part security* (MAPsec). In comparison to IPsec, Mapsec protects messages on the application layer.

Both IPsec and MAPsec should, according to [5] and [4], offer following protection: *data integrity, data origin authentication, anti-replay protection*, and *confidentiality*. In addition, IPsec should offer *limited protection against traffic flow analysis*. Nonetheless, the attack presented below does not violate any of these properties. We found it while assuming that the messages sent between S and H are encrypted and then integrity protected through a message authentication code by long-term keys shared between S and H. The *encrypt-then-mac* scheme is indeed the principle used in both IPsec and MAPsec.

3.2 Session-Mixup Attack against Authentication Data Response

We consider, as usual, an adversary who is in full control of the messages sent between instances of the roles of user U, serving network S, and home network

H. We assume that H acts as a trusted third party. When S sends an authentication data request to H for authentication parameters of U, the authentication data response by H to S is bound to U as it includes message components that are generated under the long-term key shared between H and U. However, S cannot verify this (even though we assume authenticated encryption of the messages between H and S), as S does not know the key shared between the user equipments and H. There should certainly be some mechanism for S to associate an authentication response to the correct U if there is no attacker around. But as such a mechanism is not specified in the AKA protocols, we do not model them as part of the message parts that are protected by the authenticated encryption. We present a scenario in which an inside attacker may take advantage of this and we omit, due to space restrictions, the outside attack scenario[5].

An *Inside* Attack. In this scenario we consider an attacker A who is a subscriber U of H. Say U' is another subscriber of H who is honest. If A knows the $IMSI'$ of U', which A can learn either by listening on the network or by deploying a device called *imsi catcher*, then A can execute the attack that is depicted in Figure 2 without U' even being present. In this attack scenario, A does not need to be able to intercept messages sent over the base stations. The attacker sends out two user identity responses: $IMSI'$ and his own subscriber identity $IMSI$. Then S will run two concurrent AKA sessions, one for U and one for U', and sends two authentication data requests to H. When H sends the authentication data responses for S and U, then adversary A redirects this message such that it is mistaken by S as the response by H for S and U', while A blocks the authentication data response that H generated for S and U'. Notice that this *session mixup* can be created by the attacker without breaking any cryptographic primitive and does generally not violate the specifications. Next the attacker redirects the messages sent by S intended for U' to U. So U correctly receives the user authentication request containing message components that were generated by H for U (and S). Therefore, attacker A, who is registered as U, can generate the correct response and relay it to to S such that S believes that the response was generated by U'. The other session that S opened for U' is halted by A; it cannot be completed because A does not know the keys that U' shares with H. Anyhow, A can impersonate U' to S. Furthermore, the attacker and S share a session key; it was in fact generated by H for U and S. At the same time, S believes that this session key was generated by H for S and U'. Therefore, the attacker is able to execute subsequent communication steps and use the derived keys to use the wireless service provided by S on behalf of U'. [6] S will bill H for the service that attacker A received on U' behalf, and H will bill U'.

[5] Which can, however, be easily derived from the inside attack.

[6] The attack is not fended off by the use of TMSIs. And the attacker's job is simplified in practice if multiple authentication vectors are sent at once.

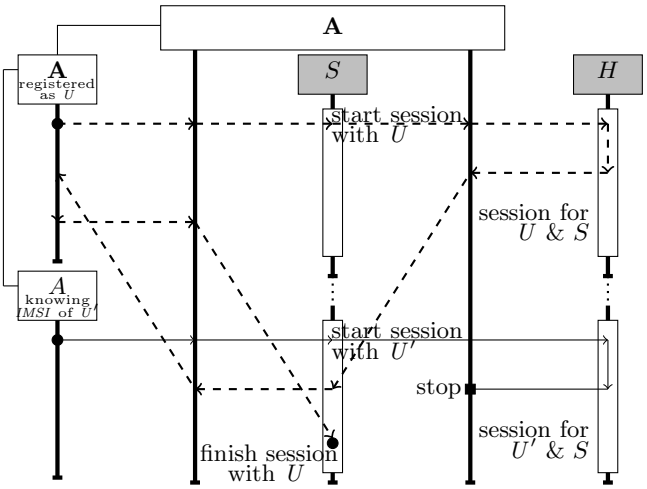

Fig. 2. Message flow of an *inside* attack against UMTS and LTE AKA (not showing the user id request). The attacker impersonates honest user U' to S and shares the session key(s) with S, without U' being involved.

3.3 The GSM Subscriber Identity Authentication Protocol

The GSM SIA protocol [9,8] is the 2G predecessor of UMTS AKA. It suffers from the same design flaw as UMTS and LTE AKA: there is no proper binding of the response sent by the home network (called *Authentication Vector Response*) to the corresponding request. Therefore, the attack of Figure 2 could also be deployed against GSM SIA. However, the case of GSM SIA is different: The specifications [9,8] are only concerned about adversaries that attack the radio path, *i.e.* the connection between user equipment and base stations, while completely neglecting other connections. It does not violate the GSM specifications if there is no protection of the authentication vector response or if the session key is transmitted in the clear by the home network. An attacker that is able to listen on the connections within the core network may not need to resort to the session-mixup attack to successfully violate GSM security as he may easily obtain the session keys. However, GSM operators that would like to protect the connection between home and serving networks, *e.g.* with MAPsec, need to be extremely careful so that message parts that prevent a session-mixup attack are sufficiently protected.

3.4 Possible Corrections

The UMTS/LTE AKA (and GSM SIA) protocols can easily be safeguarded if S is enabled to determine, even under active attacks, for which user *IMSI* a response by H was generated. We present two approaches to correcting the UMTS/LTE AKA (from which the analogous corrections of the GSM ISA can be immediately derived).

The AKA protocol can be protected against active attackers when it is slightly modified by computing and adding a value $f(IMSI, X)$ to the authentication data response, where $f(.)$ is some function, which S is able to compute and which satisfies some injectivity properties (*e.g.* f may be a hash function), and X some value known to S. Therefore, the authentication data transfer between H and S for U is changed to

$$S \longrightarrow H : \{\!\{IMSI, SNid\}\!\}_{K_{sh}}$$
$$H \longrightarrow S : \{\!\{f(IMSI, X), RAND, AUTN, XRES, Skey\}\!\}_{K_{sh}}$$

where $\{\!\{.\}\!\}_{K_{sh}}$ represents authenticated encryption under a long-term key K_{sh} shared between S and H. For instance one could choose $f(IMSI, X) \equiv IMSI$.

As an alternative fix, S could generate a fresh request identifier n_S, *e.g.* a fresh nonce, and include it in the authentication data request for U. The corresponding response must then include a function g (computable by S and with some injectivity properties) over this nonce and possibly some other data X known to S. In that case, the authentication data transfer should be modified to the challenge-response exchange

$$S \longrightarrow H : \{\!\{n_S, IMSI, SNid\}\!\}_{K_{sh}}$$
$$H \longrightarrow S : \{\!\{g(n_S, X), RAND, AUTN, XRES, Skey\}\!\}_{K_{sh}}$$

where $\{\!\{.\}\!\}_{K_{sh}}$ represents again authenticated encryption under a long-term key K_{sh} shared between S and H. For instance, $g(., X)$ could be the identity on nonces.

3.5 Feasibility of Real-World Attacks

Our attacks against the specifications of UMTS and LTE AKA work even if messages between S and H are encrypted as well as integrity protected by a message authentication code, which is what IPsec and MAPsec are doing. Although the UMTS and LTE AKA protocols are flawed, there are various scenarios in which real-world implementations of UMTS/LTE AKA could be immune to our attacks.

IPsec protects the TCP layer data. This alone does not prevent the attacks above (because IPsec would typically use the same session key for authentication data requests for both U and U'). But if, in addition to using IPsec, S uses different ports to send its requests to H, then the port numbers are appended to the sender/receiver addresses and become part of the protected TCP data. Therefore, they could be used by S to assign the responses by H correctly to each user. However, the specifications for UMTS and LTE do not detail how concurrent IPsec sessions are managed. We note that the AKA protocol is also likely run on top IPsec and other protocols, e.g. the diameter protocol [16]. If such protocols handle sessions properly and the used session identifier are protected by IPsec, then the session-mixup attack is fended off.

Likewise, MAPsec can also be used in combination with a certain way of managing sessions that prevents our attacks. In [2], which is the *implementation (stage-3) specification* for MAP, the use of an *invoke ID* is mentioned that is part of the authentication data request and the corresponding response and needs to be *unique for each serving network*. If a serving network S uses a separate invoke ID for each request, then S could assign each response correctly to the corresponding request and our attacks would no work. But again, the specifications are not detailed enough on how the invoke id is used in concurrent sessions, and using the same invoke ID for several sessions is not ruled out.

Whether actual implementations of UMTS/LTE AKA follow the strategy of combining IPsec or MAPsec with using unique ports, invoke IDs or session IDs for concurrent authentication data requests is unknown to us. While this seems to be a very natural way to implement session handling with IPsec and MAPsec, it does not seem to be required by the specifications, and therefore some implementations of UMTS/LTE AKA may indeed be vulnerable in the real-world.

Furthermore, for intra-domain connections (and in the GSM case), operators can implement their proprietary solutions instead of using IPsec or MAPsec. Therefore, such systems may currently be vulnerable to our session mix-up attack as well, even if the implementations were guided by [5,4] and offer the list of secrutiy properties that we restated in Section 3.1.

4 Conclusions and Future Work

We present a flaw in the specifications of the UMTS and LTE AKA (and GSM SIA) protocols with rather serious consequences. An inside attacker may authenticate as another honest subscriber to a serving network, and use the wireless services on his behalf. We suggest corrections to the protocols and we are in the process of using the tool CryptoVerif in order to verify entity authentication and key secrecy properties for the corrected UMTS and LTE AKA protocols [19].

We question whether it is prudent practice to make the security of the UMTS and LTE AKA protocols implicitly reliant on a specific way how IPsec or MAPsec (or other protocols on top which the AKAs are executed) should be implemented. Instead we believe that it would be preferable to strengthen the AKA protocols directly by making the binding of the home network's authentication data response for an intended user explicit in the specifications of the UMTS and LTE AKAs. Even if it turns out that real-world implementations of UMTS and LTE AKA that use IPsec or MAPsec are immune against our attack, we are convinced that the uncovered flaw still provides a valuable lesson to network domain operators who would like to protect their core networks' communication with proprietary solutions (e.g., in the case of GSM, or for IP-based intra-domain connections in the case of UMTS and LTE).

For future work, we are interested in exploring, ideally in cooperation with 3GPP and mobile network operators, to what extend real-world systems are

vulnerable to our attack. We would also like to expand our analysis scenarios of the protocol execution that are not covered in the present work, *e.g.* the scenarios that are related to the use of TMSIs and sequence numbers.

Acknowledgements. We thank Valtteri Niemi and Steve Babbage for helpful discussions.

References

1. 3GPP TS 33.102. 3G Security; Formal Analysis of the 3G Authentication Protocol, http://www.3gpp.org/ftp/Specs/html-info/33902.html
2. 3GPP TS 29.002. Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Mobile Application Part (MAP) specification, http://www.3gpp.org/ftp/Specs/html-info/29002.html
3. 3GPP TS 33.102. LTE; 3G Security; Security Architecture, http://www.3gpp.org/ftp/Specs/html-info/33102.html
4. 3GPP TS 33.200. 3G Security; Network Domain Security (NDS); Mobile Application Part (MAP) application layer security, http://www.3gpp.org/ftp/Specs/html-info/33200.html
5. 3GPP TS 33.210. LTE; 3G Security; Network Domain Security (NDS); IP network layer security, http://www.3gpp.org/ftp/Specs/html-info/33210.html
6. 3GPP TS 33.310. LTE; Network Domain Security (NDS); Authentication Framework (AF), http://www.3gpp.org/ftp/Specs/html-info/33310.html
7. 3GPP TS 33.401. LTE; 3GPP System Architecture Evolution (SAE); Security Architecture, http://www.3gpp.org/ftp/Specs/html-info/33401.html
8. 3GPP TS 42.009. Digital cellular telecommunications system (Phase 2+); Security Aspects, http://www.3gpp.org/ftp/Specs/html-info/42009.html
9. 3GPP TS 43.020. Digital cellular telecommunications system (Phase 2+); Security related network functions, http://www.3gpp.org/ftp/Specs/html-info/43020.html
10. Arapinis, M., Mancini, L.I., Ritter, E., Ryan, M.: Formal Analysis of UMTS Privacy. CoRR, abs/1109.2066 (2011), http://arxiv.org/abs/1109.2066
11. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14), Cape Breton, Nova Scotia, Canada, pp. 82–96. IEEE Computer Society (June 2001)
12. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14), Cape Breton, Nova Scotia, Canada, pp. 82–96. IEEE Computer Society (June 2001)
13. Blanchet, B.: A Computationally Sound Mechanized Prover for Security Protocols. In: IEEE Symposium on Security and Privacy, Oakland, California, pp. 140–154 (May 2006)
14. Blanchet, B.: A Computationally Sound Mechanized Prover for Security Protocols. IEEE Transactions on Dependable and Secure Computing 5(4), 193–207 (2006); Special issue IEEE Symposium on Security and Privacy 2006
15. Dolev, D., Yao, A.: On the security of public-key protocols. IEEE Transactions on Information Theory 2(29), 198–208 (1983)
16. IETF. Diameter Base Protocol RFC 3588 (September 2003), http://www.ietf.org/rfc/rfc3588.txt

17. International Telecom Union. ICT Indication Database (2011), http://www.itu.int/ITU-D/ict/statistics/
18. Meyer, U., Wetzel, S.: A man-in-the-middle attack on UMTS. In: Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe 2004), Philadelphia, PA, USA, pp. 90–97 (2004)
19. Mjølsnes, S.F., Tsay, J.-K.: Compuational Security Analysis of the UMTS and LTE Authentication and Key Agreement Protocols. CoRR, abs/1203.3866 (2012)
20. Zhang, M., Fang, Y.: Security analysis and enhancements of 3GPP authentication and key agreement protocol. IEEE Transactions on Wireless Communications 4(2), 734–742 (2005)

Blind 384-bit Digital Signature Scheme

Alexandr Moldovyan[1], Nikolay Moldovyan[1], and Evgenia Novikova[2]

[1] Laboratory of Cryptology, St. Petersburg Institute for Informatics and Automation (SPIIRAS), 39, 14th Liniya, Saint-Petersburg, Russia
nmold@mail.ru
[2] Laboratory of Computer Security Problems, St. Petersburg Institute for Informatics and Automation (SPIIRAS), 39, 14th Liniya, Saint-Petersburg, Russia
novikova@comsec.spb.ru

Abstract. The blind digital signature protocols play important role in e-commerce applications. In this paper the new blind digital signature scheme with 384-bit signature length is proposed. The latter is achieved by using finite subgroup of the multiplicative group of the finite ring of residues modulo n, where n is a product of two large primes. It is shown that proposed signature satisfies unlinkability and unforgeability properties.

Keywords: blind digital signature scheme, two-dimension-cyclicity group, difficult problem, factorization problem, discrete logarithm problem.

1 Introduction

In electronic voting systems and e-commerce requirement of the user anonymity is one of the most important requirements. Blind digital signatures are one of the cryptographic tools that can provide anonymity for users. The technology of the blind signature was first proposed by Chaum [1]. The properties of the blind signatures are : i) the signer cannot read the document during process of signature generation; ii) the signer cannot correlate the signed document with author of the message. There exist different implementations of the blind signature protocol based on discrete logarithm problem and factorization problem [2-6]. Known digital signature schemes based on discrete logarithm problem produce 4L-bit signature in case of L-bit security (i.e. the breaking of such schemes demands implementation of the 2L operations of multiplications modulo p.), while factorization-based digital signature schemes usually produce sufficiently signature with sufficiently greater length [1, 7]. Recently new approach for constructing efficient short signatures based on factorization problem was proposed [8, 9] however the used verification procedure is based on the two-level exponentiation operation and does not suits well to constructing the blind signature schemes. Short signatures are needed in environments with space and bandwidth constraints, for example, mobile environments. In this paper a new protocol of short 3L-bit blind signature scheme based on factorization problem is proposed. The shortening of the signature length is achieved by using subgroup of multiplicative group of a finite ring of residues modulo a composite number n that is difficult for factoring, like the RSA

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 77–83, 2012.

modulus. The specific feature of the used subgroup is its two-dimension cyclic structure. In paper [10] the digital signature schemes defined over finite n-dimension cyclic groups are described, however implementation of the blind signature protocols with short signatures has not been considered.

The rest of the paper is organized as follows. In the next section the procedure of the two-dimension cyclic group construction is described. In the section 2 the brief description of the used hard problem is given, the description of the proposed blind signature scheme and its correctness is given in section 3 and 4, respectively.

2 The Used Hard Problem

To construct the blind signature protocol with signature length of 384 bits computations in the subgroup of multiplicative group R_n^* of finite ring R_n are used, where n is the product of the two strong primes q and p of the length $|q| \approx |p| \approx 1232$ bits. Parameters q and p are the secret values and have specific structure: $p = N_p r^2 + 1$ and $q = N_q r^2 + 1$, where N_p and N_q are two large even numbers, containing a big prime factor, and r is a 128-bit prime. It could be shown that the multiplicative group R_n^* of a ring R_n is generated by the basis that consists of two elements. This assertion follows from the fact that the value of the generalized Euler phi function $L(n)$ is smaller than the value of the Euler phi function for the same number n:

$$\varphi(n) = (q-1)(p-1) = \mathrm{GCD}(q-1, p-1)\mathrm{LCM}(q-1, p-1) =$$

$$= \mathrm{GCD}(q-1, p-1)L(n) \geq r^2 L(n),$$

where GCD denotes to the greatest common divisor, LCM denotes to the least common multiple.

In the proposed DSS scheme we use the primary subgroup Γ of r^2 order of the group R_n^*. It is generated by the two basis elements α and β of the order r, for which we have $\alpha^i \neq \beta^j \ \forall i, j \in \{1, 2, ..., r\}$. All elements of the subgroup Γ except identity element have order r. The following procedure to generate basis elements α and β is proposed.

1. Choose the random positive integer b that is greater than 1 and less than n.
2. Compute values $\gamma = \mathrm{L}(n)/r$ and $z = b^{\gamma} \bmod n$.
3. If $z \neq 1$ holds, then take z as α (or β) otherwise repeat steps 1-3.

The correctness of the proposed procedure is easy to prove. Indeed, if $z \neq 1$ holds for the generated number z than the equation $z = bL(n)/r \bmod p$ also holds, and therefore according to the generalized Fermat theorem $z^r \equiv bL(n) \equiv 1 \bmod n$, i.e. the order of the number z equals r. (It is known that if $z^r \equiv 1 \bmod n$ holds the order of z divides number r. Since the number r is prime divisor of the value $L(n)$ then r is the order of some numbers modulo n). When implementing twice this procedure the two random numbers of an order r modulo n could be obtained. The probability that these two numbers

belong to one cyclic subgroup is determined by ratio of quantity of non-identity elements of the cyclic prime order r subgroup to the quantity of all r order elements containing in the group R_n^*. Consider that the group R_n^* contains a primary subgroup of an order r^2 generated by two elements of an order of r then this primary subgroup has $r^2 - 1$ elements of an order of r [10]. Therefore the mentioned above probability equals to $r/(r^2 - 1) \approx 1/r \approx 2^{-128}$. This probability can be reduced to the value of $\approx 2^{-256}$ if when generating numbers α and β the following modified procedure is used.

1. Choose the random positive integer b that is greater than 1 and less than n.
2. Compute values $\gamma = L(n)/r^2$ and $z = b^\gamma \bmod n$.
3. If $z \neq 1$ and $\alpha' = z^r \bmod n \neq 1$ (or $\beta' = z^r \bmod n \neq 1$) hold, then take $\alpha'^r \bmod n$ (or $\beta'^r \bmod n$) as α (or β) otherwise repeat steps 1 to 3.

The probability reduction is achieved because number of the order r^2 is generated firstly, afterwards it is raised to the power r and then the computed value is taken as α (or β). Since generated numbers α' and β' of the order r^2 belong to different cyclic subgroups Γ_{p^2} the numbers α and β also belong to different cyclic subgroups. Then the probability $\Pr(\alpha', \beta' \in \Gamma_{p^2})$ that numbers α' and β' belong to one cyclic subgroup equals to ratio of the quantity of the elements of the order r^2 lying in one cyclic subgroup to the quantity of all r^2-order elements containing in R_n^*. Considering the existence of the primary subgroup in R_n^* generated by two basis elements of order r^2, and using the formulas [10] for evaluating the quantity of elements of the given order in primary groups, it is possible to receive the following estimation of the probability:

$$\Pr(\alpha', \beta' \in \Gamma_{p^2}) = \frac{r(r-1)}{r^2(r^2-1)} \approx \frac{1}{r^2} \approx 2^{-256},$$

where the size of the value r is $|r|$=128 bits.

Thus, the second procedure for generation of the random numbers α и β is preferable as it allows reducing probability that two generated elements belong to the same cyclic subgroup in 2^{128} times. Due to the insignificance of the probability it is possible to neglect it and therefore omit resource consuming procedure of check if the generated numbers α и β belong to the same cyclic subgroup.

The security of the proposed blind DSS protocol is based on the difficulty of Discrete Logarithm Problem (DLP) modulo a composite hard factorisable number n. It is similar to the randomized DSS based on the difficulty of the DLP in finite groups with multi-dimension cyclicity [10]. If attacker wants to forge the signature of the legal user it is enough to calculate x and w using values of the public key y, α and β.

We consider two variants of the practical application of the designed protocol. In the first variant the compound number is the part of the public key, and every user generates its own unique value n, the devisors of which are the secret key of the user. In the second variant n is the secret parameter and is produced by Trusty Center that deletes secret devisors immediately after generation of parameter n. In the second

variant the length of public key is shorter however the change of parameter n will cause regeneration of all public keys. In the rest of the paper we consider the usage of the first variant of the protocol. The secret key is presented by the tuple (p, q, x, w), where p, q – divisors of the n and x, w – random 128 bit numbers ($x < r$, $w < r$). The public key is presented by tuple (n, r, α, β, y), where element y is calculated as follows $y = \alpha^x \beta^w \bmod n$.

3 The Proposed Protocol of Blind Signature

The basic idea of the blind signature protocols is that the person that signs some electronic message M doesn't know the content of the message M. Thus there are two requirements to the protocol 1) the signer can't get access to the content of the signing message; 2) later after the signature generation the signer can't link signed message to its author. The last requirement is known as the requirement of anonymity (untracability). The existing variants of blind signature protocol are constructed using known cryptographic algorithms that use following hard problems: 1) factorization problem of numbers of a type of $n = pq$, where p and q – two strong primes; 2) discrete logarithm problem modulo prime p; 3) discrete logarithm on the elliptic curve. The proposed protocol is based on difficulty of the finding discrete logarithm modulo a composite integer.

The blind signature protocol is constructed on the basis of the simple signature generation procedure. The basic DSS is described below.

The Basic DSS Generation Procedure

1. The signer generates pair of random numbers k and t ($1 < k < r$ и $1 < t < r$) and then computes parameter $R = \alpha^k \beta^t \bmod n$.
2. Then the signer calculates the values $H = F_H(M)$ and $E = F'_H(R||M) \bmod r$, where F_H and F'_H are some specified secure hash-functions having size 256 and 128 bits, respectively; M – message to be signed.
3. The signer calculates rest elements of the signature S and U as follows: $S = (k + xE) \bmod r$ and $U = (t + wE) \bmod r$.

The result of the described procedure is the 384-bit digital signature presented by the triple of the 128-bit numbers (E, S, U). The description of the proposed blind protocol is presented below. We denote user who wants to sign messages and initiates the protocol as requester while the signing entity is denoted as signer.

The Blind DSS generation Procedure

1. The signer generates random one-time disposable secret key that consists of pair of numbers k and t ($1 < k < r$ and $1 < t < r$), then computes $\overline{R} = \alpha^k \beta^t \bmod n$ and sends the value $\overline{R}$ to the requester.
2. The requester calculates $H = F_H(M)$. Then he generates the triple of the random "blinding" parameters ε, μ and τ that lie in the interval $(1, q)$ and calculate

$R = H\overline{R}^{\varepsilon} y^{\mu}\alpha^{\tau} \bmod n$, $E = F'_H(R \| M) \bmod r$ and $\overline{E} = \varepsilon^{-1}(E + \mu) \bmod r$. If $E = 0$, then requester repeats the procedure from generation of the blinding parameters. The value $\overline{E}$ is the first element of the blind signature to the message *M*. The requester sends $\overline{E}$ to the signer.

3. The signer computes the rest elements of the blind signature $\overline{S}$ and $\overline{U}$ as follows: $\overline{S} = (k + x\overline{E}) \bmod r$, $\overline{U} = (t + w\overline{E}) \bmod r$, then he sends them to the requester.
4. The requester "unblinds" parameters $\overline{S}$ and $\overline{U}$, computing thus the second and third elements of the signature *S* and *U*, respectively: $S = \varepsilon\overline{S} + \tau \bmod r$ and $U = \varepsilon\overline{U} \bmod r$.

The result of the described procedure is the 384-bit digital signature presented by the triple of the 128-bit numbers (*E*, *S*, *U*). The signature verification procedure doesn't depend on what type of signature generation (usual or "blind") was used. This corresponds to the requirement of the indistiguishability of the signature generation procedure type. The signature verification procedure includes the following steps.

1. Compute the values $F_H(M) = H$, $\widetilde{R} = Hy^{-E}\alpha^{S}\beta^{U} \bmod n$ and $\widetilde{E} = F'_H(M, \widetilde{R}) \bmod r$.
2. Compare values $\widetilde{\mathrm{E}}$ and E. If $\widetilde{\mathrm{E}}$=E holds, then the digital signature is valid.

It is easy to notice that according to the equations used on the step 3 of the protocol the elements of the generated blind signature $(\overline{E}, \overline{S}, \overline{U})$ satisfy equation $\overline{R} = y^{-\overline{E}}\alpha^{\overline{S}}\beta^{\overline{U}} \bmod n$.

4 Discussion of the Correctness, Anonymity, and Security

The correctness of the developed blind DSS can be proven by substitution of correctly created values of signature elements $(\overline{E}, \overline{S}, \overline{U})$ in the signature verification procedure. This substitution gives the following:

$$\widetilde{R} = Hy^{-E}\alpha^{S}\beta^{U} \bmod n = Hy^{-\varepsilon\overline{E}+\mu}\alpha^{\varepsilon\overline{S}+\tau}\beta^{\varepsilon\overline{U}} \bmod n = Hy^{-\varepsilon\overline{E}}y^{\mu}\alpha^{\varepsilon\overline{S}}\alpha^{\tau}\beta^{\varepsilon\overline{U}} \bmod n =$$
$$= H(y^{-\overline{E}}\alpha^{\overline{S}}\beta^{\overline{U}})^{\varepsilon} y^{\mu}\alpha^{\tau} \bmod n = H\overline{R}^{\varepsilon} y^{\mu}\alpha^{\tau} \bmod n = R \Rightarrow$$
$$\widetilde{E} = F'\left(M, \widetilde{R}\right) = F'(M, R) = E$$

Since for digital signature (*E*, *S*, *U*) produced in accordance with blind signature protocol the equation $\widetilde{\mathrm{E}}$=E holds, then signature (*E*, *S*, *U*) passes verification procedure. It is easy to show that signature produced in accordance with the usual signature protocol described above also satisfies verification equation.

It is necessary to show that proposed protocol of the blind signature also provides anonymity of the user (requester) if the signer formed blind signatures to numerous electronic messages and send them to different users. Indeed the original signature elements (*E, S, U*) to some document *M* can be bind to any "blind" signature elements using blinding parameters ε, μ and τ, computed as follows: $\varepsilon = U / \overline{U} \bmod r$, $\tau = S - U\overline{S} / \overline{U} \bmod r$ and $\mu = U\overline{E} / \overline{U} - E \bmod r$. Since the values of blinding parameters in protocol are determined randomly the signer has no assumption on requester identity for given generated signature (*E*, *S*, *U*).

It's worth noticing that every signature (*E*, *S*, *U*) generated according to the usual signature generation procedure can be interpreted as blind signature. Indeed, from the signature verification equation $R = Hy^{-E}\alpha^{S}\beta^{U} \bmod n$ follows $\overline{R} = R / H = y^{-E}\alpha^{S}\beta^{U} \bmod n$. The latter directly shows the possibility of such interpretation. Due to this possibility any two signatures can be linked by three random values as blind signature and usual valid signature calculated on the basis of this blind signature are generated to some message. This fact proves that malefactor can't use previously generated blind signatures to form new valid signature, because if such possibility exists then according to the described interpretation it would be possible to produce valid signature knowing certain number of valid signatures.

Thus the security of the proposed blind signature scheme is determined by security of the basic usual digital signature scheme DSS and security analysis of it is reduced to the security analysis of basic DSS, which can be performed as follows. Since in the proposed DSS the randomization parameter *R* is generated before the has value $E = F'_H(R\|M) \bmod r$ is computed it is possible to apply the technique [11,12] for the formal proof of the randomized DSS security. That technique is based on the possibility to force the forgery program to use the same value of the signature randomization parameter $R = H\alpha^{k}\beta^{t} \bmod n$ to produce two different signatures. In the formal security proof it is supposed that two copies of the forgery program are executed on to different computers using the same sequence of random bits that are used to make choices at various points in the work of the programs. For one of the programs the hash function F'_H is suddenly changed for F''_H one gets two different hash values computed using the same value $R = H\alpha^{k}\beta^{t} \bmod n$: $E = F'_H(R, M)$ and $E'' = F''_H(R, M)$. For the proposed DSS the forgery programs yield the output signatures (E, S, U) and (E'', S'', U'') such that

$$R = Hy^{-E}\alpha^{S}\beta^{U} = Hy^{-E''}\alpha^{S''}\beta^{U''} \bmod n \Rightarrow y = \alpha^{\frac{S''-S}{E-E''}\frac{U''-U}{E-E''}}\beta \bmod n,$$

i.e. the forgery algorithm can be applied to solve the DLP modulo *n*. Taking into account the reductionist security proof for blind signature protocol the latter has the same order of the security as the difficulty of the DLP modulo *n* which is not less than difficulty of factoring *n* [3] and in the case of 2464-bit value of *n* it has the order of the 2^{128} multiplications mod *n*.

5 Conclusions

In this paper we proposed new 384-bit blind signature scheme possessing 128-bit security which is based on difficulty of the discrete logarithm problem modulo a composite number such that factoring it is hard. Due to reducibility of the mentioned discrete logarithm problem to both the factorization problem and the discrete logarithm problem modulo prime number it is possible to state that the described cryptoscheme is based in a certain sense on the factorization problem. Further work includes performance analysis of the proposed scheme. The described approach uses the public key with specific structure that allows shortening of the signature length in the blind signature protocol. However the proposed approach can be adopted for constructing usual and blind multisignature schemes with short signature. Further study of this problem represents an interesting independent task.

References

1. Chaum, D.: Blind signature for untraceable payments. In: Advances in Cryptology (CRYPTO 1982), pp. 199–203. Plenum Press, New York (1983)
2. Tahat, N.M.F., Shatnawi, S.M.A., Ismail, E.S.: A New Partially Blind Signature Based on Factoring and Discrete Logarithms. J. of Mathematics and Statistics 4(2), 124–129 (2008)
3. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography, p. 780 . CRC Press, Boca Raton (1997)
4. Boldyreva, A.: Efficient Threshold Signature, Multisignature and Blind Signature Schemes Based on the Gap-Diffi-Hellman-Group Signature Scheme. LNCS, vol. 2139, pp. 31–46, Springer, Heidelberg (2003)
5. Camenisch, J.L., Piveteau, J.-M., Stadler, M.A.: Blind Signatures Based on the Discrete Logarithm Problem. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 428–432. Springer, Heidelberg (1995)
6. Moldovyan, N.A., Moldovyan, A.A.: Blind Collective Signature Protocol Based on Discrete Logarithm Problem. J. of Network Security 11(2), 106–113 (2010)
7. Moldovyan, N.A.: Blind Signature Protocols from Digital Signature Standards. Int. J. of Network Security 13(1), 22–30 (2011)
8. Takagi, T.: A fast RSA-type public-key primitive modulo pkq using hensel lifting. IEICE Transactions E87-A(1), 94–101 (2004)
9. Moldovyan, N.A.: An approach to shorten digital signature length. Computer Science Journal of Moldova 14(3(42), 390–396 (2006)
10. Moldovyan, N.A.: Short Signatures from Difficulty of Factorization Problem. Int. J. of Network Security 8(1), 90–95 (2009)
11. Moldovyan, N.A.: Fast Signatures Based on Non-Cyclic Finite Groups. Quasigroups and Related Systems 18, 83–94 (2010)
12. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13, 361–396 (2000)
13. Koblitz, N., Menezes, A.J.: Another Look at Provable Security. J. Cryptology 20, 3–38 (2007)

RABAC: Role-Centric Attribute-Based Access Control

Xin Jin[1], Ravi Sandhu[1], and Ram Krishnan[2]

[1] Institute for Cyber Security & Department of Computer Science
[2] Institute for Cyber Security & Dept. of Elect. and Computer Engg.
University of Texas at San Antonio
xjin@cs.utsa.edu, {ravi.sandhu,ram.krishnan}@utsa.edu

Abstract. Role-based access control (RBAC) is a commercially dominant model, standardized by the National Institute of Standards and Technology (NIST). Although RBAC provides compelling benefits for security management it has several known deficiencies such as role explosion, wherein multiple closely related roles are required (e.g., attending-doctor role is separately defined for each patient). Numerous extensions to RBAC have been proposed to overcome these shortcomings. Recently NIST announced an initiative to unify and standardize these extensions by integrating roles with attributes, and identified three approaches: use attributes to dynamically assign users to roles, treat roles as just another attribute, and constrain the permissions of a role via attributes. The first two approaches have been previously studied. This paper presents a formal model for the third approach for the first time in the literature. We propose the novel role-centric attribute-based access control (RABAC) model which extends the NIST RBAC model with permission filtering policies. Unlike prior proposals addressing the role-explosion problem, RABAC does not fundamentally modify the role concept and integrates seamlessly with the NIST RBAC model. We also define an XACML profile for RABAC based on the existing XACML profile for RBAC.

Keywords: NIST-RBAC, attribute, XACML, access control.

1 Introduction and Motivation

Role-based access control (RBAC) [12,26] is a commercially successful and widely used access control model. Access permissions are assigned to roles and roles are assigned to users. Roles can be created, modified or disabled with evolving system requirements. Since the first formalizations [26] it has been recognized that traditional formulations of RBAC are inefficient in handling fine grained access control. RBAC can accommodate fine grained authorizations by dramatically increasing the number of distinct roles with slightly different sets of permissions. However, this solution incurs significant cost of correctly assigning permissions to large numbers of roles. For instance, consider the familiar doctor-patient problem. In a hospital, a doctor is only allowed to view the record of his own patients. In the NIST RBAC model [12], a doctor role needs to be defined for each patient.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 84–96, 2012.

Thus, the number of roles will be dramatically increased while they share mostly the same permissions. Anecdotal information indicates that in practice organizations work around these limitations in ad hoc ways. The research community has also proposed several ad hoc extensions to RBAC (see section 2).

Recently Kuhn et al [23] announced a NIST initiative to unify and standardize various RBAC extensions by integrating roles with attributes, thereby combining the benefits of RBAC and attribute-based access control (ABAC) to synergize the advantages of each. An informal review of ABAC concepts is provided in Karp et al [22]. Even with the relative immaturity of ABAC formal models the NIST approach is a promising avenue for injecting the benefits of ABAC into RBAC and vice versa. We note that there are access control proposals that go beyond attributes such as [14,21]. However, we are motivated by the NIST ongoing initiative in extending RBAC through attributes, so models which go beyond ABAC are beyond our scope.

Kuhn et al [23] identify three alternatives for integrating attributes into RBAC as follows.

- **Dynamic Roles.** The first option uses user and context attributes to dynamically assign roles to users. It is similar to attribute-based user-role assignment [4]. This does help with automated user-role assignment to the myriad roles arising from role explosion, but does not address the corresponding role-permission assignment explosion (which has been considered in a recent model [19]). Context attributes have been studied in the literature [9,10,11].
- **Attribute Centric.** In this option roles are simply another attribute of users [7,20]. There is no permission-role assignment relationship. This method largely discards the advantages of RBAC which are well demonstrated and mature [15].
- **Role Centric.** The general idea in the third option is that the maximum permissions available in each session are determined by the roles activated, which can be further reduced based upon attributes. However, Kuhn et al [23] do not elaborate on this option or provide details about this approach. Moreover, to our knowledge, there are no published formal models in the literature corresponding to this option.

Our central contribution is to develop a formal model for the role-centric approach for the first time. We propose the role-centric attribute-based access control (RABAC) model which extends the NIST RBAC model with permission filtering policies. RABAC is a more convenient term otherwise identical to "RBAC-A, role-centric" in [23]. RABAC overcomes role explosion without fundamentally modifying RBAC. In particular, RABAC integrates seamlessly with the NIST RBAC model thereby offering a path for practical deployment. We also establish feasibility of implementation by providing an XACML profile for RABAC based on the existing standard XACML profile for the NIST RBAC model.

The rest of this paper is as follows. Section 2 discusses related work. Section 3 develops RABAC along with its formal definition and functional specifications.

Section 4 defines the XACML profile for RABAC and presents implementation example. Section 5 concludes the paper.

2 Related Work

The role explosion problem, wherein multiple closely related roles need to be defined to achieve fine-grained access control, has been recognized since the early days of RBAC, predating publication of the NIST RBAC model [12]. There has been considerable previous work on extending RBAC to avoid role explosion. Giuri [18] introduced the concepts of parameterized privileges and role templates to restrict a role to access a subset of objects based on the instantiated parameters. Other similar proposals include parameterized role [3,17], conditional role [6], object sensitive role [13] and attributed role [28]. The above proposals change the fundamental process of role-permission assignment as permissions assigned to roles can only be determined when a role is instantiated or activated. There is a lack of accompanying administrative models for these extensions in such context and they do not fit into the existing administration models such as [25]. Compared with roles in the NIST RBAC model, these extensions increase the complexity of role mining and engineering, which is the costliest component of RBAC [16].

Numerous other extensions of RBAC have been proposed [15]. We briefly mention a few here. TrustBAC [8] incorporated the advantages of both RBAC and credential based access control models. But only user attribute trust level is considered. A family of extended RBAC models called role and organization based access control (ROBAC) models were proposed and formalized in [29]. However, it is not designed for access control within the same organization. Kumar et al [24] extended RBAC by introducing the notions of role context and context filters. However, context filters are applied only during the process of defining roles.

3 RABAC Model

In this section, we present the RABAC model as an extension of the NIST RBAC model. The model is first discussed informally and then formally defined in two parts similar to NIST RBAC model: reference model and functional specification.

3.1 Model Overview

The RABAC model is informally depicted in figure 1. It fully incorporates the NIST RBAC model and adds the following new elements: user attributes (UATT), object attributes (OATT) and permission filtering policy (PFP). We give a brief overview of these new elements below.

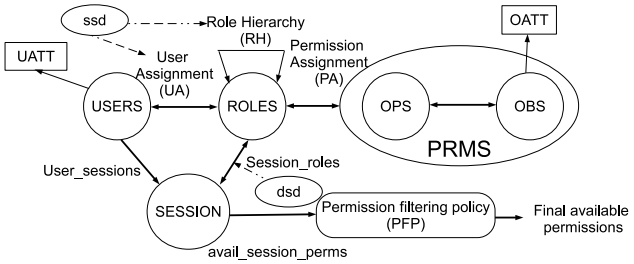

Fig. 1. RABAC Model

Attributes are functions which take certain entities and return values for defined properties of that entity (user or object).[1] Each user and object is associated with a finite set of attributes. Examples of user attributes are Department, Title and Specialization. Examples of object attributes are Type and Status. The range of each attribute is represented by a finite set of atomic values. For example, the range of Department is a set of all department names in the organization. Additionally we allow attributes to be set-valued. For instance, a set-valued Department attribute would allow a user to belong to multiple departments. Each attribute can either be atomic or set-valued from its declared range. Every attribute must be declared to be either atomic or set-valued.

The **Permission Filtering Policy (PFP)**, as suggested by its name, constrains the available set of permissions based on user and object attributes. It is depicted conceptually in figure 2. The avail_session_perm function, as defined by NIST RBAC model, gives the permission set associated with the roles activated in a given session. In RABAC the avail_session_perm function represents the maximum permission set available in a session. These permission sets are further constrained by filtering policy. The security architect specifies a set of filter functions $\{F_1, F_2, F_3 \ldots F_n\}$ for this purpose. Each filter function is a boolean expression based on user and object attributes. The $TargetFilter$ function maps each object to a subset of the filter functions. This mapping is based on the attributes of the object via attribute expressions called *conditions* which determine whether or not each filter function is applicable. The applicable filter functions are invoked one by one against each of the permissions in avail_session_perm. If any of the functions return FALSE, the permission is blocked and removed from the available permission set for this session. At the end of this process, we get the final available permission set. It should be noted that this description specifies the net result. Various optimizations can be used so long as the net result is as indicated.

With the newly defined PFP component, we are able to modify the logical approach for defining packages of functional components in the NIST RBAC model [12] as shown in figure 3. RABAC adds the dashed rectangle at the last

[1] More generally, attributes can be associated with other entities including sessions, environment, system, etc. User and object attributes suffice for purpose of RABAC.

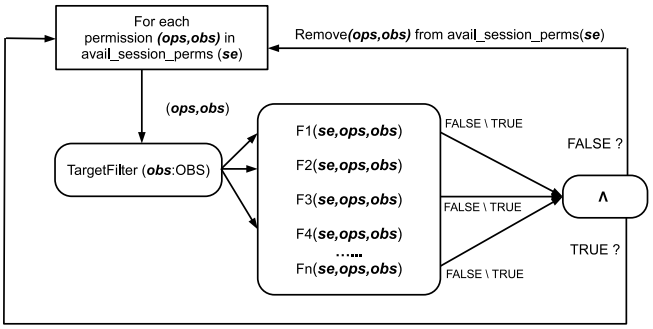

Fig. 2. Permission Filtering Process

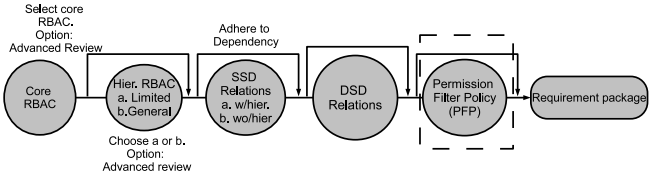

Fig. 3. Methodology for Creating Functional Packages

stage. This indicates that PFP can be integrated into each of the RBAC model components independently.

3.2 RABAC Reference Model

The basic sets and functions in the NIST RBAC model are shown in table 1. These sets and functions will also apply to RABAC. We define additional sets and functions for RABAC in table 2. UATT is a set of attribute functions for the existing users (i.e., USERS). Each attribute function in UATT maps a user to a specific value. This could be atomic or set valued as determined by the type of the attribute (as specified by $attType$). We specify similar sets and functions for objects. The notation used here for attributes is adapted from [20]. FILTER is a set of boolean functions defined by the security architects. The F_i are applied to sessions to constrain permissions associated with that session (discussed below).

The permission filtering process is configured in three steps. As illustrated in the first part of table 3, security architects firstly define each filter function F_i in terms of user and object attributes by means of the language LFilter (defined below). Security architects also need to select a subset of the filter functions that apply to an object. This is done by the $TargetFilter$ function which requires specification of a boolean condition based on object attributes for each filter function F_i. As shown in the second part of table 3, there are n such conditions, one for each F_i. Each condition is defined using the language LCondition (defined below). For an object, the $TargetFilter$ function is illustrated in the third part of

Table 1. NIST RBAC Sets and Functions used in RABAC

- USERS, ROLES, OPS, and OBS (users, roles, operations, and objects);
- PRMS = $2^{(\text{OPS} \times \text{OBS})}$, the set of permissions;
- SESSIONS, the set of sessions;
- user_sessions(u: USERS) $\rightarrow 2^{\text{SESSIONS}}$, the mapping of user u onto a set of sessions;
- avail_session_perms(s: SESSIONS) $\rightarrow 2^{\text{PRMS}}$, the permissions available to a user in a session.
- PA $\subseteq$ PRMS $\times$ ROLES, a many-to-many mapping permission-to-role assignment;
- assigned_permissions(r: ROLES) $\rightarrow 2^{\text{PRMS}}$, the mapping of role r onto a set of permissions;

Table 2. Additional Sets and Functions of RABAC

- UATT and OATT represent finite sets of user and object attribute functions respectively.
- For each *att* in UATT $\cup$ OATT, Range(*att*) represents the attribute's range, a finite set of *atomic* values.
- attType: UATT $\cup$ OATT $\rightarrow$ {set, atomic}. Specifies attributes as set or atomic valued.
- Each attribute function maps elements in USERS and OBS to atomic or set values.

$$\forall ua \in \text{UATT}.\, ua : \text{USERS} \rightarrow \begin{cases} \text{Range(ua)} & \text{if attType}(ua) = \text{atomic} \\ 2^{\text{Range(ua)}} & \text{if attType}(ua) = \text{set} \end{cases}$$

$$\forall oa \in \text{OATT}.\, oa : \text{OBS} \rightarrow \begin{cases} \text{Range(oa)} & \text{if attType}(oa) = \text{atomic} \\ 2^{\text{Range(oa)}} & \text{if attType}(oa) = \text{set} \end{cases}$$

- FILTER = {F_1, F_2, F_3, ... F_n} is a finite set of boolean functions.
  For each $F_i \in$ FILTER. F_i: SESSIONS $\times$ OPS $\times$ OBS$\rightarrow$ {T, F}.

table 3. It evaluates each condition$_i$ based on the object's attributes to determine whether or not the filter function F_i is applicable. Thus it selects a subset of the filter functions applicable for any specific object.

The languages LFilter and LCondition are defined by adopting the common policy language (CPL) from [20] as shown in table 4. CPL defines the logical structure but is not a complete language. It is required to specify the non-terminal symbols *set* and *atomic* to build complete instances of CPL. LFilter, the language used to specify each filter function F_i, is an instance of CPL where *set* and *atomic* are as follows.

set::= setua (sessionowner(*se*)) | setoa(*obs*) | ConsSet
atomic::= atomicua (sessionowner(*se*)) | atomicoa(*obs*) | ConsAtomic
setua $\in$ {ua | ua $\in$ UATT $\wedge$ attType(ua) = set }
atomicua $\in$ {ua | ua $\in$ UATT $\wedge$ attType(ua)= atomic }

Table 3. Permission Filtering for RABAC

1. Permission filtering policy.
Language LFilter is used to specify each filter function F_i(*se*:SESSIONS, *ops*:OPS, *obs*:OBS) in FILTER, where *se*, *ops* and *obs* are formal parameters.

2. Conditions.
For each $F_i \in$ FILTER there is a condition$_i$ which is a boolean expression specified using language LCondition.

3. TargetFilter is a function which maps each object to its applicable filter functions as a set. It is illustrated with the pseudo code shown as follows:

```
TargetFilter(obs:OBS)
{
    filter := {};
    condition1: filter := filter ∪ F1;
    condition2: filter := filter ∪ F2;
    ...
    conditionn: filter := filter ∪ Fn;
    return filter;
}
```

Where F_1, F_2 ... $F_n \in$ FILTER and *obs* is formal parameter.

Table 4. Common Policy Language

$\varphi ::= \varphi \wedge \varphi | \varphi \vee \varphi | (\varphi) | \neg\varphi | \exists$ x $\in$ set.$\varphi | \forall$ x $\in$ set.$\varphi |$ set setcompare set | atomic $\in$ set | atomic atomiccompare atomic
setcompare ::= $\subset$ | $\subseteq$ | $\nsubseteq$
atomiccompare ::= $<$ | $=$ | $\leq$

setoa $\in$ {oa | oa $\in$ OATT $\wedge$ attType(oa) = set }
atomicoa $\in$ {oa | oa $\in$ OATT $\wedge$ attType(oa) = atomic }

ConsSet and *ConsAtomic* are constant sets and atomic values. *se* and *obs* are formal parameters of each filtering function. LFilter use the attributes of the involved user and object. Thereby, LFilter is able to constrain permissions dynamically based on various relationships between user and object attributes. We define the *sessionowner* function to return the owner of a session as follows.

sessionowner(*se*:SESSIONS) = u such that *se*$\in$user_sessions(u)

In the above definition, user_sessions(u: USERS) is already defined in the NIST RBAC model to return the sessions for a given user. LCondition, the language for specifying conditions, is an instance of CPL where *set* and *atomic* are as follows.

set::= setoa(*obs*) | ConsSet
atomic::= atomicoa(*obs*) | ConsAtomic

Each condition can only refer to the attributes of the object *obs* being accessed. *setoa* and *atomicoa* are the same as in LFilter.

3.3 Functional Specification

Our definitions of functional specifications for RABAC are based on those already defined in NIST RBAC model. The key extensions of this model focus on access decisions. Thus, we redefine the **CheckAccess** function from NIST RBAC and define a new function called **FilteredSessionPerm**. We specify these functions in table 5. Function **FilteredSessionPerm** returns final available permissions for each specific session. Function **CheckAccess** is used to check each request (ops, obs).

Table 5. Functional Specifications

Functions	**Updates**

FilteredSessionPerm
(se: SESSIONS)
	perset = avail_session_perm(se);
	For each (ops, obs) ∈ perset **do**
	if TargetFilter(obs) = {} break;
	For each function ∈ TargetFilter(obs) **do**
	if ¬function(se, ops, obs)
	perset = perset \ {(ops, obs)}; **break;**
	return perset;

CheckAccess (se: SESSIONS, ops: OPS, obs: OBS, result: BOOLEAN)	result = ((ops, obs)∈FilteredSessionPerm(se));

4 XACML Profile for RABAC

XACML [1] is a standard language for specifying attribute based access control policy. Because of its reputation, considerable work has been done for XACML in implementing RBAC as well as its administration model [27]. XACML profile for RBAC [5] has been defined to guide implementing RBAC via XACML. For the purpose of demonstrating implementation feasibility of RABAC, we show that RABAC can be easily implemented in XACML. Specifically, we propose a XACML profile for RABAC based on that for RBAC. We then give a specific implementation example for this profile.

4.1 Proposed Profile

The standard XACML RBAC profile is limited to core and hierarchical RBAC. Our RABAC profile is similarly limited. We will only discuss those components of the standard XACML RBAC profile that need to be changed for RABAC. The RABAC profile is guided by the following.

– Permission Filtering Policies (PFP) are stored in a separate file from permission and role policy files for ease of administration.

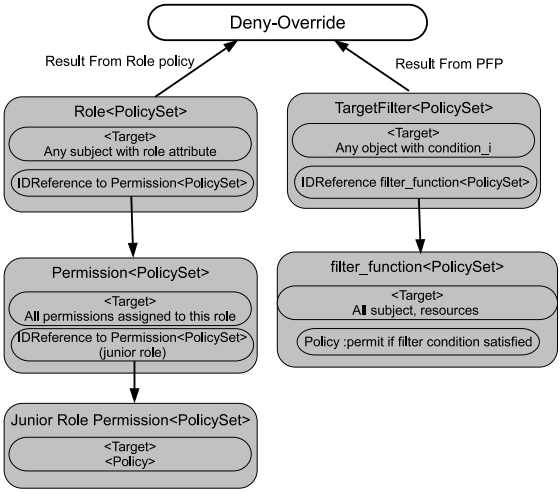

Fig. 4. Part of Proposed XACML Profile for RABAC

- The result of role policy and PFP policy may be different. We need policy combination algorithm which gives deny if and only if PFP returns deny (Note that only positive permissions are defined for role policy in NIST RBAC model). Otherwise, the final result is permit.
- The result from different filter functions upon the same group of objects should be deny-override. Thereby if any one of them returns false, the final result for PFP will be false.

In light of these observations, we design an extension where the PDP (Policy Decision Point) loads one more kind of policy files for PFP components in addition to the role policy file as shown in figure 4. The implementation for role and permission policy set remains the same. To implement PFP, a $TargetFilter$ <PolicySet> should be defined for each condition in the $TargetFilter$ function defined in the model. Conditions in $TargetFilter$ are implemented with *target* tag in XACML policy. Each $TargetFilter$ <PolicySet> contains policy references to actual $filter\_function$ <Policy>. Each reference represents a filter function defined in the model. The role policy and PFP policy may return different results. Since PFP is used only to reduce permissions there should not be PFPs that evaluate to permit. Thus, the role policy returns permit while the PFP policy may return two kinds of result $NotApplicable$ (no policy specified) or $Deny$ (not allowed). We can determine that policy combining algorithm should be deny-override. The request is with the same format as that in XACML profile for RBAC except that the XACML subject is associated with multiple attributes in addition to *role*.

Table 6. RABAC Configuration for Doctor-Patient Problem

1. Basic sets and function
UATT={doctorof, uproj} OATT={type, recordof, oproj}
attType(doctorof)= attType(uproj) = attType(oproj)= set
attType(type)= attType(recordof)= atomic
Range(uproj) = Range(oproj)={proj1, proj2, proj3 ... }
Range(type)= {PatientRecord, AuthorizedDoc ...}
Range(doctorof)= Patient
Patient is all patients maintained by the hospital, Patient⊆U.
Range(recordof)= U
FILTER= {FPatient, FAuthorized}
2. Permission filtering policy

```
FPatient(se: SESSIONS, o: OBS, read)
{
    recordof(o)∈doctorof(sessionowner(se));
}
FAuthorized(se: SESSION, o: OBS, read)
{
    ( ∃ proj1 ∈ oproj(o). ∃ proj2 ∈ uproj(sessionowner(se)).proj1=proj2 )∧
    (8:00≤time(sessionowner(se)) ∧ time(sessionowner(se)) ≤ 17:00) ∧
    device(sessionowner(se)) ∈ { set of hospital certified devices }
}
TargetFilter(o: OBS)
{
    filter = {};
    case type(o) = PatientRecord: filter = filter ∪ FPatient;
    case type(o) = AuthorizedDoc: filter = filter ∪ FAuthorized;
    return filter;
}
```

4.2 Example

We show the usage of our model in the doctor-patient problem in collaborative hospitals. The scenario is: *Doctor*, *Patient* and $VisitDoc$ are roles in each hospital. *Doctor* are allowed to read their *Patients*' record at any time. $VisitDoc$ are only allowed to read authorized documents which are revealed for collaboration purpose with other hospitals. The request will only be approved during working hours made from any hospital certified devices. In addition, visiting doctors from other hospital are only allowed to view authorized documents pertaining to the projects they participate in. We present the configuration in RABAC in table 6. The elements are to be added to original RBAC solution. In traditional RBAC, a $VisitDoc$ role for each collaborative project should be defined. As new projects are created and accomplished, $VisitDoc$ roles have to be created and deleted. In addition, roles for each project are only different in the permissions regarding the specific projects. In our solution, a general $VisitDoc$ role is defined to be able to read all authorized projects documents. Then simple filtering policy can be specified in a straightforward manner (shown below). Thus, the role needed

to be defined in traditional RBAC is the same as the number of projects while only one is needed in RABAC. Note that if the hospital requirements changes, e.g. a visit doc can read all authorized documents in his department, the role-permission and user-role relationship need to be changed in RBAC while such change is not needed in RABAC. Rather we need to change the filtering policy in RABAC, which in this case would simply delete the corresponding filtering policy.

Following the above RABAC XACML profile we have implemented the forementioned doctor-patient problem based on SUN's XACML implementation [2]. As per the standard RBAC XACML profile the role policy is straightforward. The TargetFilter <PolicySet> defines a policy for filtering access on *PatientRecord* and *AuthorizedDoc*. We take the *PatientRecord* as an example. The target is all patient records and there is a reference to the corresponding filter_function<Policy>. This policy defines a deny rule for reading patient records and the rule takes effect if resource (i.e., patient record) owner does not belong to the *doctorof* attribute value of a subject. One technical problem with this implementation is that *string-not-equal* is not natively embedded into XACML standard. Thus, we need to define this function which is straightforward and not explicitly shown here. An abbreviated portion of the XACML code for FPatiendRecord is shown below (role policy is the same as RBAC XACML profile and policy file for FAuthorized is similar).

XACML Code for PFP in Example

```
<Policy PolicyId="PFPPatiendRecord" RuleCombiningAlgId="deny-overrides">
  <Target></Any Subject>
    <Resources><!--Any PatientRecord--></Resources> </Any Action>
  </Target>
  <Rule RuleId="ReadRule" Effect="Deny">
    <Target><Any Subject Resource/>
      <Actions><Action><ActionMatch MatchId="string-equal">
         <AttributeValue DataType="string">read</AttributeValue>
           <ActionAttributeDesignator DataType="string"
             AttributeId="action:action-id"/>
           </ActionMatch></Action></Actions> </Target>
    <Condition FunctionId="string-not-equal">
         <Apply FunctionId="string-one-and-only">
         <SubjectAttributeDesignator DataType="string"
         AttributeId="doctorof"/></Apply>
      <Apply FunctionId="string-one-and-only">
       <ResourceAttributeDesignator DataType="string"
           AttributeId="owner"/></Apply>
   </Condition>
   </Rule>
</Policy>
```

5 Conclusion and Future Work

In this paper, we proposed RABAC, a novel extension to the NIST RBAC model in an effort to address the role explosion problem of RBAC without modifying significant components of RBAC model and retaining the static relationships between roles and permissions. It is the first model to integrate roles and attributes using the role centric approach identified by Kuhn et al [23]. RABAC

integrates roles and attributes in a flexible and reliable manner. In particular, we define an independent component called the permission filtering policy (PFP) adding to the existing components of the NIST RBAC model. We also extend the functional specification of the NIST RBAC model and XACML profile for RBAC. Our solution essentially retains the administration convenience of RBAC while ensuring flexibility and scalability without role explosion.

There are several interesting directions for future work. Formal analysis of tradeoffs between roles and attributes may provide practically useful insights and results. The language CPL, which is used for specifying the filtering function as well as conditions in $TargetFilter$ functions, can be extended to leverage the power of XACML as these functions can be expressed through XACML policy files.

Acknowledgment. The authors are partially supported by grants from AFOSR MURI and the State of Texas Emerging Technology Fund.

References

1. OASIS, Extensible access control markup language (XACML), v2.0 (2005).
2. Sun's XACML implementation, http://sunxacml.sourceforge.net/index.html
3. Abdallah, A.E., Khayat, E.J.: A Formal Model for Parameterized Role-Based Access Control. In: Formal Aspects in Security and Trust (2004)
4. Al-Kahtani, M.A., Sandhu, R.: A model for attribute-based user-role assignment. In: ACSAC (2002)
5. Anderson, A.: XACML profile for role based access control (RBAC). Technical Report Draft 1, OASIS (February 2004)
6. Bao, Y., Song, J., Wang, D., Shen, D., Yu, G.: A Role and Context Based Access Control Model with UML. In: ICYCS (2008)
7. Chadwick, D.W., Otenko, A., Ball, E.: Implementing Role Based Access Controls Using X.509 Attribute Certificates. IEEE Internet Computing (2003)
8. Chakraborty, S., Ray, I.: TrustBAC: integrating trust relationships into the RBAC model for access control in open systems. In: SACMAT (2006)
9. Cirio, L., Cruz, I.F., Tamassia, R.: A Role and Attribute Based Access Control System Using Semantic Web Technologies. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2007, Part II. LNCS, vol. 4806, pp. 1256–1266. Springer, Heidelberg (2007)
10. Covington, M.J., Long, W., Srinivasan, S., Dev, A.K., Ahamad, M., Abowd, G.D.: Securing context-aware applications using environment roles. In: SACMAT (2001)
11. Covington, M.J., Sastry, M.R.: A Contextual Attribute-Based Access Control Model. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1996–2006. Springer, Heidelberg (2006)
12. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Richard Kuhn, D., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. on Infor. and Sys. Sec. (2001)
13. Fischer, J., Marino, D., Majumdar, R., Millstein, T.: Fine-Grained Access Control with Object-Sensitive Roles. In: Drossopoulou, S. (ed.) ECOOP 2009. LNCS, vol. 5653, pp. 173–194. Springer, Heidelberg (2009)

14. Fong, P.W.L.: Relationship-based access control: protection model and policy language. In: CODASPY (2011)
15. Fuchs, L., Pernul, G., Sandhu, R.S.: Roles in information security-A survey and classification of the research area. Computers & Security (2011)
16. Gallagher, M.P., O'Connor, A.C., Kropp, B.: The economic impact of role-based access control. In: Planning report 02-1, NIST, (March 2002)
17. Ge, M., Osborn, S.L.: A design for parameterized roles. In: DBSec (2004)
18. Giuri, L., Iglio, P.: Role templates for content-based access control. In: Proc. of the Second ACM Workshop on RBAC. ACM (1997)
19. Huang, J., Nicol, D., Bobba, R., Huh, J.H.: A Framework Integrating Attribute-based Policies into RBAC. In: SACMAT (2012)
20. Jin, X., Krishnan, R., Sandhu, R.: A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. In: DBSec (2012)
21. Kalam, A.A.E., Benferhat, S., Miege, A., Baida, R.E., Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y., Trouessin, G.: Organization based access control. In: POLICY (2003)
22. Karp, A.H., Haury, H., Davis, M.H.: From ABAC to ZBAC: the evolution of access control models, In: Tech. Report, HP Labs (2009)
23. Richard Kuhn, D., Coyne, E.J., Weil, T.R.: Adding Attributes to Role-Based Access Control. IEEE Computer 43(6), 79–81 (2010)
24. Kumar, A., Karnik, N., Chafle, G.: Context sensitivity in role-based access control. SIGOPS Oper. Syst. Rev. 36(3), 53–66 (2002)
25. Sandhu, R., Bhamidipati, V., Munawer, Q.: The ARBAC97 model for role-based administration of roles. ACM Trans. on Info. and Sys. Sec. (1999)
26. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
27. Xu, M., Wijesekera, D., Zhang, X., Cooray, D.: Towards Session-Aware RBAC Administration and Enforcement with XACML. In: POLICY (2009)
28. Yong, J., Bertino, E., Toleman, M., Roberts, D.: Extended RBAC with role attributes. In: 10th Pacific Asia Conf. on Info. Sys. (2006)
29. Zhang, Z., Zhang, X., Sandhu, R.: ROBAC: Scalable role and organization based access control models. In: IEEE TrustCol (2006)

Trust-Aware RBAC

Vladimir Oleshchuk

Department of ICT, University of Agder
PB 509, N-4898 Grimstad, Norway
vladimir.oleshchuk@uia.no

Abstract. In this paper we propose a trust-aware enhancement of RBAC (TA-RBAC) that takes trustworthiness of users into consideration explicitly before granting access. We assume that each role in the framework is associated with an expression that describe trustworthiness of subjects required to be able to activate the role, and each subject (user) has assigned trustworthiness level in the system. By adding trustworthiness constraints to roles we enhance system, for example, with more flexible ability to delegate roles, to control reading/updating of objects by denying such operations to those subjects that violate trustworthiness requirements.

1 Introduction

Over the years, Role-Based Access Control (RBAC) has established itself as a generalized approach for handling access control in computer systems and differs from traditional identity based access control models in that it takes advantage of the concept of role relations [11, 10]. For these models, access to data and resources are based on the organizational activities and responsibilities, or roles, which users possess in a system. In RBAC, a user's ability to access computer resources (objects) is determined by the user's association with roles and by these roles' permissions to perform operations on objects. Usually roles correspond to different job functions within an organization. Job functions are associated sets of permissions, which can be seen as expression of trustworthiness of role holder within an organization.

Different access control models to support various organizational security policies have been proposed over the years. The first model that supported integrity protection of resources was Biba Model developed by Kenneth J. Biba in 1977 [5]. This model describes a set of access control rules designed to ensure data integrity. The idea is that subjects on lower integrity levels are not permitted to modify (corrupt) objects on higher integrity levels (known as "no write up" rule). Correspondently, subjects on higher integrity levels can be corrupted by accessing objects on lower integrity levels (known as "no read down" rule). This model can be considered as one of the first models dealing with trustworthiness (also implicitly). According to [6], "integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change".

Bell-LaPadula model was proposed to support protection of confidentiality [3]. This model describes access rules to protect confidentiality of resources.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 97–107, 2012.

Assuming that subjects are assigned clearance levels and objects are assigned classification levels the model's rule support no write down and no read up rules. However these rules also prevent effective communication.

Role Based Access Control (RBAC) Model has been found to be quite useful and has drawn a lot of research interest over the last fifteen years. It was recently defined as NIST/ANSI Standard [2]. Traditional RBAC considers user to role as well as role to permission assignments to be static in nature with respect to space and time. However it was observed that the context-aware access should play a more active role in access decision process. For example, in mobile applications, spatial context plays an increasingly important role both in defining and enforcing more elaborated security policies since in many applications locations of participants should directly influence access control decisions [1, 7, 9, 17–19, 22].

In recent years some new extensions of RBAC that use notion of trust have being proposed [4, 8, 24, 25]. Trust-aware access control models are more suitable for decentralized, multi-centric systems with dynamic population of users where traditional access models do not work well. One of the benefit of trust awareness (not considered yet in existing extensions of RBAC) is an ability to provide history-based solutions.

In this paper we propose to enhance the discrete trust paradigm with more elaborated multi-level trust paradigm based on notion of opinion from subjective logic. In this new trust-enhanced model, trust levels of subjects (human users or software agents acting on users behalf), roles and objects (data, programs, processes) are expressed as opinions about their trustworthiness determined by history of interactions between subjects (users) and objects. We define such opinions in the framework of subjective logic [14, 15].

The rest of the paper is organized as follows. In Section 2 we provide brief review of related work. We introduce notation and notions of subjective logic in Section 3. Then we present our trust-enhanced RBAC model in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

Recent years many context-aware extensions of RBAC were proposed. Most of them considered specifically access wth respect of time and location.

In [17, 18], authors extend the RBAC model by specifying spatial restrictions on permissions assigned to roles which enables a role to have permissions assigned to it dependent on the location. Spatial constraints on permissions assigned to a role can be beneficial when specifying the access control policy in mobile environments where the location from which a user wnats to access services is a key security parameter [19]. Authors have extended the RBAC model, and introduced a formal model that allows specifying spatial constraints on permissions associated with roles in different locations. In [9] authors propose a spatially-aware RBAC model called GEO-RBAC. In the proposed approach authors propose the notion of spatial roles which are defined as roles with spatial extents defining the boundaries of the space in which the role can be used by the

users. In this approach roles are activated based on the position of the user. Another location aware RBAC model has been proposed in [22]. Authors show how the different components of the core RBAC model are related to location, how existing operations need to be modified and what new operations are needed. They left elaboration of role hierarchies and separation of duty constraints for future work. Several authors have also proposed models that combine both spatial and temporal aspects [7, 1].

However, relatively recently some authors started to consider trust as a new context parameter. This is motivated by desire to expand RBAC models to meet security challenges posed by new application paradigms were existing models found to be inadequate (for example, for open and decentralized systems or mobile and pervasive systems). In [4] authors motivation was to provide access control model for Web services. They propose an extended, trust-enhanced version of XML-based RBAC (X-RBAC) framework that incorporates context-based access control. In the framework, the authors rely on certification provided by trusted third party assigning levels of trust to users. In [8] authors propose a trust based access control model called TrustBAC by extending the conventional RBAC model with the notion of trust levels. Users are assigned to trust levels instead of roles based on a number of factors like user credentials, user behavior history, user recommendation etc. Trust levels are numbers between -1 and $+1$ and which computes by the trust evaluation module. In [24] authors propose a trust-based RBAC model for pervasive computing systems where they adapt trust model they proposed earlier [23] to evaluate trustworthiness. Needs for delegation arise in many applications. Trustworthy delegation that does not violate security policies by allowing access only to trustworthy delegatee was considered in [25]. In [20] authors propose a framework that combines strenths of RBAC systhems and trust-management systems to deal with access control in decenralized collabarative systems. The trustworthness of subjects is determined on the base of their certified attributes.

3 Measurement of Trust: Subjective Logic

In this section, we show how to express the levels of trustworthiness in the framework of subjective logic. Following [14, 15] we first define the term opinion, denoted ω, that expresses opinion about level of trustworthiness.

Let t, d and u be such that $t + d + u = 1$ and $t, d, u \in [0, 1]$. Then a triple $\omega = \{t, d, u\}$ is called an opinion, where components t, d and u represent levels of trust, distrust and uncertainty respectively. The levels of trustworthiness are expressed by opinions. Varying these parameters, we can express different levels of trustworthiness. Expressing trustworthiness using three values instead of just one trust level provides a more adequate trust model of real world with uncertainties. These parameters are not treated equally when different opinions are combined.

The subjective logic defines a set of logical operators for combining opinions including conjunction, recommendation, and consensus. For more details related to subjective logic the reader is recommended to consult [14–16].

Let $\omega^A = \{t^A, d^A, u^A\}$ denote an opinion about trustworthiness of entity A.

Let $\omega_p^A = \{t_p^A, d_p^A, u_p^A\}$ denote an opinion of entity A about consequences for security of an action p. In context of this paper, A can be an RBAC system itself or a user, and an action p may be "activate role r". Assume that an entity A has an opinion $\omega_p^A = \{t_p^A, d_p^A, u_p^A\}$ about potential security threat of p, and an opinion $\omega_q^A = \{t_q^A, d_q^A, u_q^A\}$ about potential security threat q. Then A's opinion about consequences for security of both actions, denoted as $p \wedge q$, can be found (according to [14]) as following:

$$\omega_{p\wedge q}^A = \omega_p^A \wedge \omega_q^A = \{t_{p\wedge q}^A, d_{p\wedge q}^A, u_{p\wedge q}^A\}$$

where

$$t_{p\wedge q}^A = t_p^A t_q^A$$
$$d_{p\wedge q}^A = d_p^A + d_q^A - d_p^A d_q^A$$
$$u_{p\wedge q}^A = t_p^A u_q^A + u_p^A t_q^A + u_p^A u_q^A$$

Let A and B be two entities (RBAC systems or users). If A is a RBAC system itself, it has opinions about trustworthiness of its own users but not about users in other RBAC systems (in a federated system). Then $\omega_B^A = \{t_B^A, d_B^A, u_B^A\}$ denotes an opinion of A entity about trustworthiness of recommendations given by B. Assume B gives its recommendation to A about trustworthiness of action p in the form of its opinion ω_p^B. Assuming that an entity A does not have any direct opinion ω_p^A about p it will try to deduce some indirect opinion about trustworthiness of p, denoted ω_p^{AB}, based on recommendation given by B. For this purpose the recommendation operator $\otimes$ is used (according to [14]) as follows:

$$\omega_p^{AB} = \omega_B^A \otimes \omega_p^B = \{t_p^{AB}, d_p^{AB}, u_p^{AB}\}$$

where

$$t_p^{AB} = t_B^A t_p^B$$
$$d_p^{AB} = t_B^A d_p^B$$
$$u_p^{AB} = d_B^A + u_B^A + t_B^A u_p^B$$

In context of this work there is a need to combine independent opinions about trustworthiness of the same action. According to [16], "The consensus opinion of two possibly conflicting argument opinions is an opinion that reflects both argument opinions in a fair and equal way". Adjusting reasoning from [16] we can argue applicability of the consensus operator (defined below).

Let A and B be two entities that represent entities such as the system or users. Let $\omega^A = \{t^A, d^A, u^A\}$ and $\omega^B = \{t^B, d^B, u^B\}$ be two opinions of A and B about the same action (object, user). In the case when there are several independent opinions about the same action, subjective logic suggests to use a consensus operator $\oplus$ to combine these independent opinions. According to subjective logic, the combined consensus opinion ω based on ω^A and ω^B is defined as follows:

$$\omega = \omega^A \oplus \omega^B$$

where

$$t = (t^A u^B + t^B u^A)/(u^A + u^B - u^A u^B)$$
$$d = \left(d^A u^B + d^B u^A\right) / \left(u^A + u^B - u^A u^B\right) r$$
$$u = \left(u^A u^B\right) / \left(u^A + u^B - u^A u^B\right)$$

We define odering relation on opinions in the following way. We assume that opinion ω^A is more trustworthy than opinion ω^B, denoted $\omega^A \gg \omega^B$, if $t^A > t^B$. If $t^A = t^B$, then higher distrust value means lower corresponding uncertainty value. Assuming that decreasing uncertainty may contribute equally to both trust and distrust values, we choose opinions with higher uncertainty (and with equal trust values t^A and t^B) be more trustworthy. Formally, if $t^A = t^B$ then $\omega^A \gg \omega^B$ if $u^A > u^B$.

In the following section we describe how trustworthiness expressed as opinions can be integrated into traditional RBAC model.

4 Trust-Aware RBAC (TA-RBAC)

Informally, we assume that there is a set of roles $ROLES$ that may be assigned to users from $USERS$. Each user u may have many roles assigned at the same time. Each role r from $ROLES$ is associated with a pair $\{\omega^l, \omega^h\}$. It means that trustworthiness of a user u who are able to activate r cannot be lower than ω^l and higher than ω^h respectively.

In this section we propose a trust-aware RBAC (TA-RBAC) model that is an extension of traditional RBAC model [2]. Since the time of introduction of RBAC, various context-aware models were proposed (see Section 2). However some important features dictated by current and future real-world applications such as, for example, handling access request in proximity of specific devices [21] or taking history of behavior or actions are still not well-developed. This is the motivation behind the proposed extension.

Informally, in the proposed extension of RBAC each role r from $ROLES$ has assigned requirements on trustworthiness of users u from $USERS$ that are permitted to activate this role. It means that to have role assigned to u is not enough - it is also necessary verify that the current level of trustworthiness of u satisfies trustworthiness requirements assigned to the role r. This is an additional constraint (requirement) that may be used to take into account behavior history of u such as what roles u has activated in the past, for which purposes, from which location, when, etc. Dynamically changing trustworthiness constraints of roles provides additional constraints on ability of u to activate assigned roles, for example, in case of trust-aware separation of duties (explained in Subsection 4.3).

The proposed TA-RBAC model consists of the following five basic components: $USERS$, $ROLES$, $PRMS$, $SESSIONS$ and TRW representing the set of users, roles, permissions, sessions and opinions respectively, where TRW is a set of possible opinions about trustworthiness of users and roles. Users from $USERS$ are considered to be either humans, devices or processes operating on

behalf of other users that can access resources (services) to perform some actions. $ROLES$ describes a collection of roles defined as a set of permissions that may be guarded by trustworthiness constraints to control accessibility to resources (objects). $PRMS$ is a set of permissions to access resources/services to perform a specific action if trustworthiness of the user satisfies trust requirements. Elements of TRW is specified by means of subjective logic opinions.

4.1 Core Model

The model defines several functions and relations on the sets $USERS$, $ROLES$, $PRMS$, $SESSIONS$, TRI needed for specification and implementation of TA-RBAC. The user assignment relation UA represents the assignment of a user from $USERS$ to roles from $ROLES$. The permission assignment relation PA represents the assignment of permissions to roles based on trustworthiness of both users and services. Definition below gives formal descriptions of some important functions and relations.

Definition 1. *TA-RBAC core model consists of the following components:*

- $USERS$, $ROLES$, $PRMS$, $SESSIONS$ *and* TRW, *represent the finite sets of users, roles, permissions, sessions, and opinions respectively;*
- $PRMS = REQUESTS \times SERVICES$ *where* $REQUESTS$ *denotes all action requests users can send to services denoted as* $SERVICES$*;*
- TRW *represents trustworthiness in form of subjective logic opinions (including complete trust* $(1,0,0)$*, complete distrust* $(0,1,0)$ *and complete uncertainty* $(0,0,1)$*);*
- $UA \subseteq USERS \times ROLES$ *, the relation that associates users with roles;*
- $TRI \subseteq TRW \times TRW$ *represents the set of trustworthiness intervals, where* $(\omega_1, \omega_2) \in TRI$ *means* $\omega_1 \gg \omega_2$ *or* $\omega_1 = \omega_2$*;*
- $UT \subseteq USERS \times TRW$ *defines assignment of trustworthiness level to users;*
- $assigned\_trust\,(u : USERS) \rightarrow TRW$ *, the function mapping a user* u *into an opinion. Formally, trustworthiness of user* u *can be found as* $assigned\_trust\,(u) = \{\omega | (u, \omega) \in UT\}$*;*
- $RT \subseteq ROLES \times TRI$ *defines assignment of trustworthiness level to roles;*
- $role\_trust\_constr(r : ROLES) \rightarrow TRI$*, the function that maps a role* r *to trustworthiness interval from* TRI*. Formally, trustworthiness constraints associated with* r *can be found as* $role\_trust\_constr(r) = \{t | (r, t) \in RT\}$*.*
- $assigned\_users(r : ROLES) \rightarrow 2^{USERS}$ *, the mapping of a role onto a set of users. Formally, users assigned to role* r *can be found as* $assigned\_users(r) = \{u \in USERS | (u, r) \in UA\}$ *;*
- $assigned\_roles(u : USERS) \rightarrow 2^{ROLES}$ *, the mapping of a user* u *onto a set of roles. Formally, roles assigned to a user* u *can be found as* $assigned\_roles(u) = \{r \in ROLES | (u, r) \in UA\}$*;*
- $PA \subseteq ROLES \times PRMS$ *, the relation that defines what permissions* $PRMS$ *of a role* r *from* $ROLES$ *are available to a user with trustworthiness level suitable to activate* r*. That is* $(r, p) \in PA$ *means that if user* u *has assigned role* r *she can utilize permission* $p = (req, srv)$ *to access service* srv *when trustworthiness of* u *satisfies trust requirement of* r*.*

- *assigned_perms*$(r : ROLES, t : TRW) \rightarrow 2^{PRMS}$ *describes permissions assigned to role* r *when trustworthiness of a user* u *activating* r *satisfies trust requirements of* r*. Formally,* $assigned\_perms(r) = \{p|(r,p) \in PA\}$*;*
- *user_sessions*$(u : USERS) \rightarrow 2^{SESSIONS}$*, associate a user* u *with set of sessions;*
- *session_roles*$(s : SESSIONS) \rightarrow 2^{ROLES}$*, the mapping of session* s *to a set of roles;*
- *avail_session_perms*$(s : SESSIONS, t : TRI) \rightarrow 2^{PRMS}$ *, the permissions available in a session* s *when trust requirements satisfies* t*.*
  Formally, $avail\_session\_perms(s,t) = \bigcup_{r \in session\_roles(s)} assigned\_perms(r,t)$
- *auth_user*$(r : ROLES, t : TRI) \rightarrow USERS$ *identifies users assigned to role* r *satisfying* t*.*
- *auth_user*$(r_1, r_2, ... : ROLES) \rightarrow USERS$ *identifies users assigned to at least roles* $r_1, r_2, ..., r_k$*;*
- *auth_user*$(s : SESSIONS; r_1, r_2, ... : ROLES) \rightarrow USERS$ *identifies users that are authorized to activate simultaneously roles* $r_1, r_2, ...r_k$ *within a session* s*.*

We assume that each user u assigned an initial trust level, $init\_trust : USERS \rightarrow TRW$, and each role r from $ROLES$ has assigned trustworthiness constraints as $\left[\omega^l, \omega^h\right]$ from TRI where ω^l denotes the lowest trust level that a user u must have to be able activate r; ω^h denotes a trustworthiness level that user activating r must not exceed (in cases when it is not essential $\omega^h = (1,0,0)$, meaning highest possible trustworthiness).

4.2 Role Hierarchies

In order to extend the core TA-RBAC to Hierarchical TA-RBAC we need to define hierarchies and inheritance for roles in presence of trust describing how roles inherit permissions from their junior roles. Definition below formally introduces our solution.

Definition 2. *Relation* RH*, defines as* $RH \subseteq ROLES \times ROLES$*, is a partial order on roles, with respect to trustworthiness, called dominance relation, denoted as* $\succeq$ *, where* $r_i \succeq r_j$ *for* $r_i, r_j \in ROLES$ *means that* r_i *inherits permissions of* r_j*. Let* $role\_trust\_constr(r_j) = (\omega_j^l, \omega_j^h)$ *and* $role\_trust\_constr(r_i) = (\omega_i^l, \omega_i^h)$ *. We saying that role* r_i *inherits permissions of role* r_j *with trustworthiness constraints for user assigned to* r_i *computed as following:* $(\omega_j^l \oplus \omega_i^l, \omega_j^r \oplus \omega_i^r)$*.*

The use of the consensus operator can be argued that trustworthiness constraints (ω_i^l, ω_i^h) and (ω_j^l, ω_j^h) can be seen as independent opinions on trustworthiness constraints of r_j.

The reason for use of the consensus operator is as following. There are two independent sets of trustworthiness constraints of r_i: 1) the trusworthiness constraints on r_i independant of hierarchies, and 2) trustworthiness constraints derived from constraints of the inherited role r_j. In case of two independent

opinions the consensus operator usually applied to find a combined opnions that constitutes the new constraints. The user assigned to r_i have to satisfy native constraints of r_i to use its permissions. However when the user wants to use inherited permissions of r_j, she must satisfy constraints that are consensus between constraints of r_i and r_j. A user assigned directly to r_j have to satisfy native constraints of r_j in order to activate its permissions.

4.3 Separation of Duties

The efficiency of RBAC to enforce the principle of least privilege is partly due to ability to enforce Separation of Duties (SoD) principle. However by constraining ability of users to activate some combination of roles reduce system's usability, for example, in small organizations where number of users are small with respect to number of roles.

The trust-awareness may provide better flexibility by for example putting less constraints on highly trustful users. However, that means that the notion of SoD needs to be re-defined. We defines both Trust-aware Static SoD (TSSoD) and Trust-aware Dynamic SoD (TDSoD), where requirement to trustworthiness of a user who activates (partly) mutually exclusive roles increases. That is, two roles with assigned permissions may be partly mutually exclusive if system requires higher level of trustworthiness comparing to requirements when only each role will be activated separately.

We define trust-aware separation of duty SoD property as following set. Let $S = \{(r, \omega), ...\}$ where $r \in ROLES$, $\omega \in TRW$ and ω informally represents opinion of the RBAC system on how security sensitive activation of r is. When a user u activates a set of roles $r_1, r_2, ..., r_k$ such that from $\{(r_i, \omega_i) | i = 1, ..., k\} \subseteq S$ the requirements to trustworthiness of the user will be computed as $\omega_1 \wedge ... \wedge \omega_k$. Informally, it means that more trustworthy users are able to activate simultaneously more roles from S.

Formally, TSSoD can be defined as following.

Definition 3. *$TSSoD \in 2^{ROLES \times TRW}$ is a set of pairs (r, ω) where is a role, ω is a trustworthiness, with the property that no user can be assigned to subset of roles from TSSoD such that conjuction of opinions ω_i of assigned roles will exceed initial trustworthiness of that user. Formally, $\{(r_i, \omega_i) | i = 1, 2, ..., k\} \subseteq TSSoD \wedge \forall u \in auth\_user(r_i \, | i = 1, ..., k) \Rightarrow user\_trw(u) \gg \wedge_{i=1,2,...,k} \omega_i$*

Formally, TDSoD can be defined as following.

Definition 4. *$TDSoD \in 2^{ROLES \times TRW}$ is a collection of pairs (r, ω) where r is a role, ω is a trustworthiness, with the property that no user can activate a subset of roles from TDSoD such that conjunction of opinions ω_i of roles activated within a session will exceed initial trustworthiness of that activating user. Formally: $\{(r_i, \omega_i) \, | i = 1, 2, ..., k\} \subseteq TDSoD \wedge s \in user\_sessions(u) \Rightarrow \forall u \in auth\_user(s, r_i \, | i = 1, ..., k) \Rightarrow user\_trw(u) \gg \wedge_{i=1,2,...,k} \omega_i$*

The reason of using conjunction operator is as following. Since opinion about security implications (potential security threats) of activation role r_i is ω_i the

opinion about security implications of activation simultaniosly a set of such roles can be computed as a conjunction of coresponding opinions.

4.4 Delegation

Many authors have studied role delegation in RBAC [27, 26, 25]. In this work we propose to control the ability to delegate roles by taking into consideration trustworthiness of delegatee and trustworthiness constraints of delegated role. The idea is that a user with high degree of trustworthiness can delegate role with relatively high trustworthiness requirements to less trustful user (since combination of role constraints and user trustworthiness can decrease required trustworthiness and therefore make it available for less trustful user). Delegation is not a part of standart RBAC and it may result in security violation. However it may provide better usability.

Suppose that a user u wants to delegate her role r to another user u' (that has not this role assigned). One way to do this is to add the instance of r called r' (r' is a new instance of r which may differs from r by trustworthiness constraints) to u' by adding r' to the list of assigned roles assigned to u'. That is

$$assigned\_role(u') = assigned\_role(u') \cup \{r', (\omega^l_{r'}, \omega^h_{r'})\},$$

where the trustworthiness requirements $(\omega^l_{r'}, \omega^h_{r'})$ on this delegated role r' will be computed as combination of trustworthiness of u and constraints of r as following (since it can be seen as a recommendation of r by u to u') as following:

$$\omega^l_{r'} = \omega_u \otimes \omega^l_r$$

$$\omega^h_{r'} = \omega_u \otimes \omega^h_r$$

where ω_u denotes trustworthiness of u. The use the recommendation operator in this case because we consider delegation of role r by u to u' as a recommendation of u to the system to assign r to u'. The system uses trusworthiness of u as trustworthiness of u's recommendations and computes trustworthiness constraints for r' by taking into consideration of trustworthiness of u.

5 Conclusion

In this work we propose a novel trust-aware RBAC model (TA-RBAC). Our approach integrates trustworthiness levels expressed as opinions in subjective logic with traditional RBAC model. We have defined trust-aware role inheritance which is essential for defining Hierarchical trust-aware RBAC. By using subjective logic operations for combining independent opinions we define static and dynamic trust-aware SoD. We use recommendation operator to define trust-aware role delegation.

References

1. Aich, S., Sural, S., Majumdar, A.: STARBAC: Spatiotemporal Role Based Access Control. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part II. LNCS, vol. 4804, pp. 1567–1582. Springer, Heidelberg (2007)
2. ANSI/INCITS 359-2004. Role Based Access Control. InterNational Committee for Information Technology Standards (formerly NCITS) / 03-Feb-2004 / 56 pages
3. Bell, D.E., LaPadula, L.J.: Secure Computer Systems: Mathematical Foundations. MITRE Corporation (1973)
4. Bhatti, R., Bertino, E., Ghafoor, A.: A Trust-Based Context-Aware Access Control Model for Web-Services, Distributed and Parallel Databases (2005)
5. Biba, K.J.: Integrity Considerations for Secure Computer Systems, MTR-3153, The Mitre Corporation (April 1977)
6. Bishop, M.: Computer Security: Art and Science. Addison Wesley, Boston (2003)
7. Chandran, S.M., Joshi, J.B.D.: LoT-RBAC: A Location and Time-Based RBAC Model. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 361–375. Springer, Heidelberg (2005)
8. Chakraborty, S., Ray, I.: TrustBAC: integrating trust relationships into the RBAC model for access control in open systems. In: Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies (SACMAT 2006), pp. 49–58. ACM, New York (2006)
9. Damiani, M.L., Bertino, E., Catania, B., Perlasca, P.: Geo-RBAC: A spatially aware RBAC. ACM Trans. Inf. Syst. Secur. 10, 1–42
10. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control. Artech House (2003)
11. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC) 4(3), 224–274 (2001)
12. Ferreira, A., Chadwick, D., Farinha, P., Correia, R., Zao, G., Chilro, R., Antunes, L.: How to securely break into RBAC: The BTG-RBAC model. In: Annual Computer Security Applications Conference, ACSAC 2009, pp. 23–31 (December 2009)
13. Ferreira, A., Cruz-Correia, R., Antunes, L., Farinha, P., Oliveira-Palhares, E., Chadwick, D., Costa-Pereira, A.: How to break access control in a controlled manner. In: 19th IEEE International Symposium on Computer-Based Medical Systems CBMS 2006, pp. 847–854 (2006)
14. Jøsang, A.: An Algebra for Assessing Trust in Certification Chains. In: Kochmar, J. (ed.) Proceedings of the Networks and Distributed Systems Security, NDSS 1999 (1999)
15. Jøsang, A.: A Logic of Uncertain Probabilities, International Journal of Uncertainty. Fuzziness and Knowledge-Based Systems 9(3), 279–311 (2001)
16. Jøsang, A.: The Consensus Operator for Combining Beliefs. Artificial Intelligence Journal 142(1-2), 157–170 (2002)
17. Hansen, F., Oleshchuk, V.: Spatial role-based access control model for wireless networks. In: IEEE Vehicular Technology Conference VTC 2003, vol. 3, pp. 2093–2097 (2003)
18. Hansen, F., Oleshchuk, V.: SRBAC: A spatial role-based access control model for mobile systems. In: Proceedings of the Seventh Nordic Workshop on Secure IT Systems (Nordsec 2003), October 15-17, pp. 129–141 (2003)

19. Hansen, F., Oleshchuk, V.: Location-based security framework for use of handheld devices in medical information systems. In: Fourth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom Workshops 2006, March 13-17, pp. 564–569 (2006)
20. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society Press (2002)
21. Oleshchuk, V., Fensli, R.: Remote patient monitoring within a future 5G infrastructure. Wireless Personal Communications 57, 431–439
22. Ray, I., Kumar, M., Yu, L.: LRBAC: A Location-Aware Role-Based Access Control Model. In: Bagchi, A., Atluri, V. (eds.) ICISS 2006. LNCS, vol. 4332, pp. 147–161. Springer, Heidelberg (2006)
23. Ray, I., Ray, I., Chakraborty, S.: An interoperable context sensitive model of trust. Journal of Intelligent Information Systems 32(1), 75–104 (2009)
24. Toahchoodee, M., Abdunabi, R., Ray, I., Ray, I.: A Trust-Based Access Control Model for Pervasive Computing Applications. In: Gudes, E., Vaidya, J. (eds.) Data and Applications Security XXIII. LNCS, vol. 5645, pp. 307–314. Springer, Heidelberg (2009)
25. Toahchoodee, M., Xie, X., Ray, I.: Towards Trustworthy Delegation in Role-Based Access Control Model. In: Proceedings of the 12th International Conference on Information Security, Pisa, Italy, September 07-09 (2009)
26. Wainer, J., Kumar, A.: A fine-grained, controllable, user-to-user delegation method in RBAC. In: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies (SACMAT 2005), pp. 59–66. ACM, New York (2005)
27. Zhang, X., Oh, S., Sandhu, R.: PBDM: a flexible delegation model in RBAC. In: Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT 2003), pp. 149–157. ACM, New York (2003)

Alternative Mechanisms for Information Security

Alexander Grusho, Nick Grusho, and Elena Timonina

Institute of Informatics Problems, RAS,
Vavilova str. 44, 119333 Moscow, Russia
{grusho,eltimon}@yandex.ru, info@itake.ru

Abstract. The usage of doubtful data for information protection is considered. It is shown that by means of unauthenticity it is possible to protect confidentiality, integrity and availability. The basic methods of usage of unauthenticity are mentioned. We introduce the mathematical models and present estimations of quality for information protection with help of injection of unreliable information.

Keywords: information protection, unauthenticity of the information, intellectual noise.

1 Introduction

The method of deception of the opponent at the expense of provision to it false information is known since antique times, as for cryptography went in throughout thousands years. However the cryptography became a science, when C.Shannon defined how to evaluate that ciphers are good. Authors of this paper didn't find scientific works devoted to an estimation of quality of privacy and integrity protection by means of adding of doubtful information. In this paper ideas for quality estimations of information protection are offered when using doubtful information. In the first model of the section 2 the estimation of quality of privacy protection of information has numerical character and can be expressed through a distance between true distribution of a choice of a decision and distribution of a choice of the decision in case of adding of doubtful information. For example the Kullback-Leibler's distance [1] between these measure can be used. However the parameter λ of our model can also be chosen as the numerical characteristic of quality.

The analyst developing privacy protection by means of implementation of doubtful information, can calculate distance in this model and compare to the given threshold. If the distance is more than threshold, the analyst makes the decision that the implementation method of doubtful information can be read rather resistant. If the distance is less than threshold, the quantity of doubtful information for privacy protection should be increased.

The second model developed in section 2 allows to construct an asymptotic estimation of quality of information protection by means of adding of doubtful data as probabilities of acceptance of the false decision.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 108–113, 2012.

The second model in the section 2 is similar to models which are used in artificial intelligence with applications to sociology [3].

Information protection, as a rule, is used for protection of confidentiality, integrity and availability. Numerous literature is devoted to methods of protection of information (see, for example, [2]). However the necessity of new methods for information protection is evident.

Examples of nonconventional methods of protection are considered in the paper and the simplest estimations of the quality of these methods are carried out. In the description of methods of protection of confidentiality it is necessary to rise from level of representation of information to level of work with information.

The information circulates in a standard cycle:

information gathering - information analysis - decision-making by results of the analysis - implementation of decisions

Value of the information is defined by its utility for the analysis and decision-making. True information and correct analysis provide, as a rule, the correct decision, which helps to achieve the target object. If the information is doubtful, the confidence in correct results of the analysis doesn't present, then the decision, based on results of the analysis, maybe wrong, i.e. will not lead to the target object in view.

The main idea of the work is to add doubtful information to protect confidentiality and integrity and also estimate the quality of such protection. Of course it is not simple to generate necessary doubtful information because we should use special algorithms and sometimes artificial intellect [4].

Let's consider applications of unauthenticity of the information for protection of confidentiality and integrity. It is possible to use these methods to protect availability also.

2 Model of Unauthenticity for Confidentiality Protection

The next models give a base for numerical estimations of quality of data confidentiality protection. An analyst can defined different thresholds for these numerical estimations. These thresholds give formal definition of what we want of privacy:

- the false decision is more preferable;
- identical arguments pro's and con's the correct and false decision;
- impossibility of the proof of unauthenticity of the information used for the decision.

Let X be a set of decisions, P_0 be a distribution of probabilities on X for a choice of decisions by results of the true information analysis, P_1 be a distribution of probabilities on X for a choice of decisions on the basis of the false information analysis. Then the distribution of probabilities on X

$$P = \lambda P_0 + (1 - \lambda) P_1,$$

where $0 \leq \lambda \leq 1$, is the simplest model for a choice of decision on the basis of doubtful or incomplete information.

To introduce the artificial unauthenticity means to choose P_1 and λ. It is clear that P_1 should be different from P_0. But it may be far away from an aprior information on sense of the decision. Then λ produces likelihood for the emulation of true decision. Thus unauthenticity of the information is the absence of confidence in its correctness.

Let Y be a set of elementary fragments of the initial information, y be the concrete information for analysis carrying out, $P(x|y)$ be conditional probability of decision-making x under condition of use for this decision of the initial data y, $Q(y)$ be probability of reception of the initial information y. We will assume that it is possible to use the following formula of a total probability

$$P(x) = \sum_{y \in Y} Q(y) P(x|y).$$

Protection of confidentiality of the correct decision is realized through granting to the adversary of unreliable information. Let Q_0 and Q_1 be distributions of a choice of the initial information $y \in Y$ in true and false cases accordingly. If for receiving true distribution of the decision the formula is fair

$$P_0(x) = \sum_{y \in Y} Q_0(y) P(x|y),$$

and for the false decision the formula is fair

$$P_1(x) = \sum_{y \in Y} Q_1(y) P(x|y),$$

then according to our model about probability of a choice of the decision the formula is fair

$$P(x) = \sum_{y \in Y} \left(\lambda Q_0(y) + (1-\lambda) Q_1(y)\right) P(x|y).$$

From here simple communication between artificial adding of unauthenticity and decision distortion follows. The problem of protection of confidentiality of the information consists in the fact that the legal analyst knew and used $Q_0(y)$, and the analyst of the adversary for decision-making used distribution $Q(y)$, which in our model is in the formula

$$Q(y) = \lambda Q_0(y) + (1-\lambda) Q_1(y)$$

at the unknown right side of the equation.

For achievement of demanded effect of security it is possible to transfer or store more information. Let $y_1, ..., y_n$ be fragments of information, where the fragment y_i is true (i is known to a legal analyst as a key), other information is received according to distribution Q_1.

Thereby the massive of the received data corresponds to a choice from distribution Q', where distribution

$$Q' = \frac{1}{n}Q_0 + \frac{n-1}{n}Q_1.$$

The adversary by data $y_1, ..., y_n$ in our model can restore no more than distribution Q'. Thus expression Q' through Q_0 and Q_1 isn't known to him and $Q_0 \neq Q'$. Then the adversary chooses the decision according to distribution

$$P' = \frac{1}{n}P_0 + \frac{n-1}{n}P_1.$$

Thus given representation isn't known to the adversary. The more P_0 differs from P' the better confidentiality of the true data is protected.

Thus, artificial unauthenticity of the information can protect confidentiality of the authentic data.

We investigate the problem with the help of following simple model. We suppose that all sets are finite, $|X| = N$, $|Y| = M$. For each $y \in Y$ the number of admissible solutions $X_y \subseteq X$ is fixed. Let's consider for simplicity that for all $y \in Y$ we have $|X_y| = m$. Let the true data y_0 are received not randomly and define a set of admissible decisions X_{y_0}. All other solutions $x \in X \setminus X_{y_0}$ are inadmissible for the true data. For protection of confidentiality of the true data $y_1, ..., y_n$ randomly get out from Y, which define sets $X_{y_1}, ..., X_{y_n}$ which are subsets of X.

We offer the elementary reasonable decision rule. Let's use the finiteness of considered scheme and we will calculate frequencies of occurrence $\nu(x)$ of separate decisions in X, which integrally met in $X_{y_0}, X_{y_1}, ..., X_{y_n}$. We accept the decision $\tilde{x}$, which occurs the maximal number of times.

Let's specify determination of randomness in our model at getting of artificial data. Assume that for each $y \in Y$ the choice of X_y is random and equiprobable in X. It is equivalently to random and equiprobable sampling of a row of length of N from 0 and 1, containing exactly m units. Then the chosen data represent a random matrix from N columns and n rows. It is convenient to consider the asymptotical case of the problem if $N \to \infty, n \to \infty, M >> n + N$. Thus naturally to believe that the probability of a repeated choice of y is negligible. It means that it is possible to consider that all rows of a matrix are equiprobable and independent from each other. For simplicity we will work in conditions when $\max_{x \in X} \nu(x)$ is reached in one point. It is possible to prove that with probability tending to one there is only one point $x \in X$ that satisfy $\arg\max_{x \in X} \nu(x)$ when $N = o\left((mn)^{\frac{1}{3}}\right)$.

It is apparent that the defined probability measure is invariant for renumbering of columns. At the considered restrictions it means that distribution of a place of the single maximum of $\max_{x \in X} \nu(x)$ is equiprobable. Then asymptotically

$$P\left(\arg\max_{x \in X} \nu(x) \notin X_{y_0}\right) = 1 - \frac{m}{N}.$$

It means that at $m = o(N)$ with the probability aspires to 1, the decision which is accepted by the opponent will be unacceptable for the true data. Thus the legal analyst makes the decision, using a key - an arrangement of the true input data.

3 Model of Unauthenticity for Integrity Protection

The method of integrity control on the base of steganography is offered in this section.

Let the legal message be inserted in some legal container by means of a method of steganography [5]. The receiver gets the message from the container, and throws the container away. Let the method of steganography depend on a key known to the sender and to the receiver.

It means that the adversary has no trustworthy information about the place of the message in the container. Then the adversary is compelled to bring distortions randomly. It will damage the container with high probability. Container distortions allow to reveal possible infringements of integrity of the true message and to request its repeated transfer. Even, if the adversary repeatedly deforms new transfer, it with a high probability deforms other signs on the true message. Then by voting method the true message can be restored.

For some methods of a steganography we define estimations of quality of integrity control. Namely, mathematical expectation E of number of distorted letters in the hidden information message can be used for such estimation of quality. It is possible to consider even more difficult parameter for estimation of quality. It is probability of not distortion of the hidden message.

Let's give estimation of E for LSB method of steganography. Let the probability of distortion of one bit in case of attack on integrity is equal to p, probability to use place i for transmission of the hidden message we will designate q. Then $E = npq$, where n is the length of the text in which the hidden message is embedded. Thus, it is possible to evaluate simply quality of integrity protection when LSB method is used.

4 Methods of Inserting of Unauthenticity into Information

To bring unauthenticity into information it is possible to present any information object in the form of a set of variables. The concrete information represents a set of values of these variables. Information gathering represents reading of values of these variables. Ways of inserting of unauthenticity into the data are following:

- distortion or change of values of variables;
- liquidation of variables (absence of corresponding values in the transferred data);
- creation of a false variable (occurrence of values of a false variable creates discrepancy of the initial data more often).

For example, for identification of the hostile code in computer system the protection program should identify some events. If any of these events are absent, the program cannot accept the true decision on presence of a hostile code.

Doubtful information creation can sometimes represent a challenge. In particular, it is required, when the false information object should look quite plausibly, i.e. satisfy to a number of the logic restrictions connected with correct information. For example, if testing of programs needs true data and it is forbidden, hence, creation of the plausible data for testing is necessary.

The greatest obstacle in doubtful information creation is the process of accumulation of the information. In the process of accumulation the quantity of logic restrictions on the data increases. Then the probability of revealing of unauthenticity of the information also increases. However this problem dares by means of creation of models of information objects (for example, UML - models [6]).

5 Conclusion

The purpose of the work is to build models and create estimations of quality of nonconventional methods for information protection, such as unauthenticity creation. Doubtful information is widely used for the military purposes, however these methods also can be applied effectively in competitive struggle and for solution of other problems. Nonconventional methods allow to expand a set of protected objects. For example, when using unreliable information it is simpler to build protection for information processes. As shown in the paper, there are new mathematical models of information security.

Acknowledgements. Work is supported by Russian Foundation for Basic Research, the grant 10-01-00480.

References

1. Prokhorov, Y.V., Rozanov, Y.: A Probability theory: Science (1973) (in Russian)
2. Pieprzyk, J., Hardjono, T., Seberry, J.: Fundamentals of Computer Secyrity. Springer (2003)
3. Mikheenkova, M.A.: About principles of the formalized qualitative analysis of sociological data. J. Information Technologies and Computing Systems 4 (2009)
4. Grusho, A.A., Timonina, E.E.: Intellectual noise. J. Problems of Information Protection Computer Systems 1 (2000)
5. Grusho, A.A., Grusho, N.A., Timonina, E.E.: Some of application steganography and security of steganographic scemes. J. Problems of Information Protection Computer Systems 2 (2007)
6. Turmoils, G., Yakobson, A., Rambo, J.: UML. Classics CS. 2 izd./lanes with English: Under the general edition of prof. S. Orlova - SPb.: Peter (2006)

Enforcing Information Flow Policies by a Three-Valued Analysis

Josée Desharnais, Erwanne P. Kanyabwero, and Nadia Tawbi

Department of Computer Science and Software Engineering, Université Laval,
{josee.desharnais,nadia.tawbi}@ift.ulaval.ca,
erwamme-pamela.kanyabwero.1@ulaval.ca

Abstract. This paper presents an approach to enforce information flow policies using a three-valued type-based analysis on a core imperative language. Our analysis aims first at reducing false positives generated by static analysis, and second at preparing for instrumentation. False positives arise in the analysis of real computing systems when some information is missing at compile time, for example the name of a file, and consequently, its security level. The key idea of our approach is to distinguish between negative and may responses. Instead of rejecting in the latter cases, we type instructions with an additional type, *unknown*, indicating uncertainty, possibly preparing for a light instrumentation. During the static analysis step, the may responses are identified and annotated with the *unknown* security type, while the positive and negative responses are treated as is usually done. This work is done in preparation of a hybrid security enforcement mechanismWe prove that our type system is sound by showing that it satisfies non-interference. The novelty is the handling of three security types, but we also treat variables and channels in a special way. Programs interact via communication channels. Secrecy levels are associated to channels rather than to variables whose security levels change according to the information they store.

1 Introduction

Secure information flow analysis is a technique used to prevent misuse of data. This is done by restricting how data are transmitted among variables or other entities in a program, according to their security classes.

Our objective is to combine static and dynamic analysis. We design a three-valued type system to statically check non-interference for a simple imperative programming language. To the usual main security levels, public (or *Low*) and private (or *High*), we add a third value, *Unknown*, that captures the possibility that we may not know before execution whether the information is public or private. Standard two-valued analysis has no choice but to be pessimistic with uncertainty and hence generate false positive alarms. If uncertainty arises in the analysis, we tag the instruction in cause: in a second step, instrumentation at every such point together with dynamic analysis will allow us to head to a more precise result than purely static approaches. We get reduced false alarms, while introducing a light runtime overhead by instrumenting only when necessary.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 114–129, 2012.

The goal of a security analysis is to ensure non-interference, that is, to prevent inadvertent information leaks from private channels to public channels. More precisely, in our case, the goal is to ensure that 1) a well-typed program satisfies non-interference, 2) a program not satisfying non-interference is rejected 3) a program that may satisfy non-interference is detected and sent to the instrumentation step. Furthermore, we consider that programs have interaction with an external environment through communication *channels*, i.e., objects through which a program can get information from users (printing screen, file, network, etc.). In contrast with the work of Volpano et al. [1], variables are not necessarily channels, they are local and hence their security type is allowed to change throughout the program. This is similar to flow-sensitive typing approaches like the one of Hunt and Sands, or Russo and Sabelfeld [2,3]. Our approach distinguishes clearly communication channels, through which the program interacts and which have a priori security levels, from variables, used locally. Therefore, our definition of non-interference applies to communication channels: someone observing the final information contained in communication channels cannot deduce anything about the initial content of the channels of higher security level.

We aim at protecting against two types of flows, as explained in [4]: *explicit flow* occurs when the content of a variable is directly transferred to another variable, whereas *implicit flow* happens when the content assigned to a variable depends on another variable, i.e., the guard of a conditional structure.

The rest of this paper is organized as follows. After describing in Section 2 the programming language used, we present the type system ensuring that information will not be leaked improperly, in Section 3. The soundness of the type system is proved in Section 4. We compare our work to similar approaches in the literature in Section 5. We conclude in Section 6.

2 Programming Language

We illustrate our approach on a simple imperative programming language, a variant of the one presented by Smith [5], which we adapt to deal with 3-valued security levels.

2.1 Syntax

Let $\mathcal{V}ar$ be a set of identifiers for variables, and $\mathcal{C}$ a set of communication channel names. Throughout the paper, we use generically the following notation: variables are $x \in \mathcal{V}ar$, and there are two types of constants: $n \in \mathbb{N}$ and $nch \in \mathcal{C}$. The syntax is as follows:

$$
\begin{array}{lll}
\textit{(instructions)} & p ::= & e \mid c \\
\textit{(expressions)} & e ::= & x \mid n \mid nch \mid e_1 \ \mathbf{op} \ e_2 \\
\textit{(commands)} & c ::= & \mathbf{skip} \mid x := e \mid c_1; c_2 \\
 & & \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{end} \mid \mathbf{while}\ e\ \mathbf{do}\ c\ \mathbf{end} \mid \\
 & & \mathbf{receive_c}\ x_1\ \mathbf{from}\ x_2 \mid \\
 & & \mathbf{receive_n}\ x_1\ \mathbf{from}\ x_2 \mid \\
 & & \mathbf{send}\ x_1\ \mathbf{to}\ x_2
\end{array}
$$

Instructions are either expressions or commands. Values are integers (we use zero for false and nonzero for true), or channel names. **op** stands for arithmetic or logic binary operators on integers and comparison operators on channel names. Commands are mostly the standard instructions of imperative programs.

We suppose that two programs can only communicate through channels (which can be, for example, files, network channels, keyboards, computer screens, etc.). We assume that the program has access to a pointer indicating the next element to be read in a channel and that the send to a channel would append an information in order for it to be read in a first-in-first-out order. When an information is read in a channel it does not disappear, only the read pointer is updated, the observable content of a channel remains as it was before. Our programming language is sequential; we do not claim to treat concurrency and communicating processes as it is treated in [6,7]. We consider that external processes can only read and write to public channels. The instructions related to accessing channels deserve further explanations.

- **receive$_\mathbf{c}$** x_1 **from** x_2: stands for "receive content". It represents an instruction that reads a value from a channel with name x_2 and assigns its content to x_1.
- **receive$_\mathbf{n}$** x_1 **from** x_2: stands for "receive name". Instead of getting data from the channel, we receive another channel name, which might be used further in the program. This variable has to be treated like a channel.
- **send** x_1 **to** x_2: used to output on a channel with name x_2 the content of the variable x_1.

The need for two different receive commands is a direct consequence of our choice to distinguish variables from channels. It will be clearer when we explain the typing of commands, but observe that this allows, for example, to receive a private name of channel through a public channel [1]: the information can have a security level different from its origin's. This is not possible when variables are observable.

2.2 Semantics

The behavior of the program follows the structural operational semantics shown in Table 1. An instruction p is executed under a memory map $\mu : \mathcal{V}ar \to \mathbb{N} \cup \mathcal{C}$. Hence the semantics specifies how *configurations* $\langle p, \mu \rangle$ evolve, either to a value, another configuration, or a memory. Evaluation of expressions under a memory involves no "side effects" that would change the state of memory. In contrast, the role of commands is to be executed and change the state. Thus we have two evaluation rules: $\langle e, \mu \rangle$ leads to a value resulting from the evaluation of expression e on memory μ; this transition is designated by $\to_e$, $\langle c, \mu \rangle$ leads to a memory produced by the execution of command c on memory μ; this transition is designated by $\to$.

skip leaves the memory state unchanged. The assignment $x := e$ results in a memory identical to μ, except that its value at x is now the evaluation of e.

[1] But not the converse, to avoid implicit flow leaks.

Table 1. Structural operational semantics

(VAL)	$\langle v, \mu\rangle \rightarrow_e v,$
(VAR)	$\langle x, \mu\rangle \rightarrow_e \mu(x),$
(OP)	$\dfrac{\langle e_1, \mu\rangle \rightarrow_e v_1 \qquad \langle e_2, \mu\rangle \rightarrow_e v_2 \qquad v_1 \textbf{ op } v_2 = n}{\langle e_1 \textbf{ op } e_2, \mu\rangle \rightarrow_e n}$
(SKIP)	$\langle \textbf{skip}, \mu\rangle \rightarrow \mu$
(ASSIGN)	$\dfrac{\langle e, \mu\rangle \rightarrow_e v}{\langle x := e, \mu\rangle \rightarrow \mu[x \mapsto v]}$
(RECEIVE-VAL)	$\dfrac{x_2 \in dom(\mu) \qquad read(\mu(x_2)) = n}{\langle \textbf{receive}_\textbf{c}\ x_1 \textbf{ from } x_2, \mu\rangle \rightarrow \mu[x_1 \mapsto n]}$
(RECEIVE-NAME)	$\dfrac{x_2 \in dom(\mu) \qquad read(\mu(x_2)) = nch}{\langle \textbf{receive}_\textbf{n}\ x_1 \textbf{ from } x_2, \mu\rangle \rightarrow \mu[x_1 \mapsto nch]}$
(SEND)	$\dfrac{x_1 \in dom(\mu)}{\langle \textbf{send } x_1 \textbf{ to } x_2, \mu\rangle \rightarrow \mu, update(\mu(x_2), \mu(x_1))}$
(COND)	$\dfrac{\langle e, \mu\rangle \rightarrow_e n \qquad n \neq 0}{\langle \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ end}, \mu\rangle \rightarrow \langle c_1, \mu\rangle}$
	$\dfrac{\langle e, \mu\rangle \rightarrow_e n \qquad n = 0}{\langle \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ end}, \mu\rangle \rightarrow \langle c_2, \mu\rangle}$
(LOOP)	$\dfrac{\langle e, \mu\rangle \rightarrow_e n \qquad n = 0}{\langle \textbf{while } e \textbf{ do } c \textbf{ end}, \mu\rangle \rightarrow \mu}$
	$\dfrac{\langle e, \mu\rangle \rightarrow_e n \qquad n \neq 0}{\langle \textbf{while } e \textbf{ do } c \textbf{ end}, \mu\rangle \rightarrow \langle c; \textbf{while } e \textbf{ then } c \textbf{ end}, \mu\rangle}$
(SEQUENCE)	$\dfrac{\langle c_1, \mu\rangle \rightarrow \mu'}{\langle c_1; c_2, \mu\rangle \rightarrow \langle c_2, \mu'\rangle}$

$\textbf{receive}_\textbf{c}\ x_1 \textbf{ from } x_2$ and $\textbf{receive}_\textbf{n}\ x_1 \textbf{ from } x_2$ are semantically evaluated similarly. Information from the channel x_2 is read and assigned to the variable x_1. The distinctive feature of the rule RECEIVE-VAL is that the result of evaluation is an integer variable, while for the rule RECEIVE-NAME, the result is a channel name. Here, we introduce a generic function *read(channel)* that represents the action of getting information from a channel (eg. get a line from a file, input

from the keyboard, etc.). The content of a channel remains the same after both kind of receive.

send x_1 **to** x_2 updates the channel x_2 with the value of the variable x_1. This is done by the generic function *update(channel, information)*, which represents the action of updating the channel with some information. Note that the content of the variable x_2, that is, the name of the channel, does not change; hence μ stays the same. The content of the channel is updated after a **send**.

A conditional statement **if** e **then** c_1 **else** c_2 **end** evaluates to c_1 or c_2 depending whether e evaluates to **true** (nonzero) or **false** (zero).

For the loop rule, if the boolean condition evaluates to **false**, the loop is not entered and the memory remains the same; if the condition evaluates to *true*, the body c is executed followed sequentially by the re-execution of the **while** instruction.

If c_1 transforms memory μ into μ', then $c_1; c_2$ amounts to the execution of c_2 on μ'.

3 Security Type System

We now present the security type system that we use to check whether a program, written in the language described above, either satisfies non-interference, may satisfy it or does not satisfy it. The security types are defined as follows:

$$\begin{array}{ll} \textit{(data types)} & \tau ::= \ L \mid U \mid H \\ \textit{(instruction types)} & \rho ::= \tau \ \textit{val} \mid \tau \ \textit{chan} \mid \tau \ \textit{cmd} \end{array}$$

We consider a set of three security levels $SL = \{L, U, H\}$. This set is extended to a lattice $(SL, \sqsubseteq)$ using the following order: $L \sqsubseteq U \sqsubseteq H$ (we use freely the usual symbols $\sqsupseteq$ and $\sqsupset$). It is with respect to this order that the supremum $\sqcup$ and infimum $\sqcap$ over security types are defined. We lift this order to instruction types and maps in the trivial way and assume these operation return $\bot$ when applied to instructions of different types, e.g., $H \ \textit{chan} \sqcup H \ \textit{val} = \bot$. We also need the following weaker relation $\preceq$, defined as:

$$x \not\preceq y \ \text{ iff } \ x = H \text{ and } y = L.$$

This relation can be interpreted as "maybe $\sqsubseteq$": it is used to ensure that rejection of a program will only occur if there is a flow from H to L; it will be explained later on.

A label is added to the type of an instruction in order to indicate whether the information is an integer value, a channel name or a command. When typing a program, security types are assigned to variables, channels and commands – and to the context of execution. The meaning of types is as follows. A variable of type τ *val* has a content of security type τ; a channel of type τ *chan* can store information of type τ or lower (indeed, a private channel must have the possibility to contain or receive both private and public information). The security typing of commands is standard, but has a slightly different meaning: a command of type τ *cmd* is guaranteed to only allow flows into channels whose security types

are τ or higher. Hence, if a command is of type $L\ cmd$ then it may contain a flow to a channel of type $L\ chan$.

Our type system has two interesting properties: *simple security* applying to expressions and *confinement* applying to commands [5]. *Simple security* says that an expression e of type $\tau\ val$ or $\tau\ chan$ contains only variables of level τ or lower. Simple security ensures that the type of a variable is consistent with the principle stated in the precedent paragraph. *Confinement* says that a command c of type $\tau\ cmd$ executed under a context of type pc allows flows only to channels of level $\tau \sqcup pc$ or higher, in order to avoid a flow from a channel to another of lower security (H to L for example). Those two properties are used to prove non-interference.

Our typing rules are shown in Table 2. A *typing judgment* has the form $\Gamma, pc \vdash p : \rho, \Gamma'$, where Γ and Γ' are typing environments, mapping variables to a type of the form $\tau\ val$ or $\tau\ chan$, representing their security level; pc is the security type of the context. The program is typed with a context of type L; according to the security types of conditions, some blocks of instructions are typed with a higer context, as will be explained later. The typing judgment can be read as: within an initial typing environment Γ and a security type context pc, the command p has type ρ, yielding a final environment Γ'. When the typing environment stays unchanged, Γ' is omitted. Since the type of channels is constant, there is a particular typing environment for channel constants, named *TypeOf_Channel* that is given before the analysis. In the rules, α stands for either the label *val* or *chan*, depending on the context.

There are three operators on typing environments that we need to define: $\Gamma \dagger [x \mapsto \rho]$, $\Gamma \sqcup \Gamma'$ and $\overline{\Gamma}$. The former is a standard update, where the image of x is set to ρ, no matter if x is in the original domain of Γ or not. For the conditional rule, we need to perform a union of environments where common value variables must be given, as security type, the supremum of the two types, and where channel variables are given type U if they differ. More precisely, we extend $\sqcup$ to environments as follows: $\mathrm{dom}(\Gamma \sqcup \Gamma') = \mathrm{dom}(\Gamma) \cup \mathrm{dom}(\Gamma')$, and

$$\Gamma \sqcup \Gamma'(x) = \begin{cases} \Gamma(x) & \text{if } x \in \mathrm{dom}(\Gamma) \setminus \mathrm{dom}(\Gamma') \\ \Gamma'(x) & \text{if } x \in \mathrm{dom}(\Gamma') \setminus \mathrm{dom}(\Gamma) \\ U & \text{if } \Gamma(x) = \tau\ chan \neq \tau'\ chan = \Gamma'(x) \\ \Gamma(x) \sqcup \Gamma'(x) & \text{otherwise.} \end{cases}$$

Note that $\Gamma \sqcup \Gamma'(x)$ can return $\bot$ if Γ and Γ' are incompatible on variable x, for example if $\Gamma(x)$ is a value, and $\Gamma'(x)$ is a channel (this can only happen if Γ and Γ' come from different branches of an **if** command).

The last operator on typing environment that we need to define relates to instrumentation. We introduce a special variable *_instr* whose type (maintained in the typing environment map) tells whether or not the program needs instrumentation. If the image of *_instr* is U or H then instrumentation is needed and L otherwise. Initially, $\Gamma(\_instr) = L$. Its value is updated to U or H through the supremum operator in rule RECEIVE-NAME_S and through an operator denoted by " $\overline{\ }$ ", in rule SEND_S, defined as follows:

Table 2. Typing rules

(CHAN_S) $$\frac{\mathit{TypeOf\_Channel}(nch) = \tau}{\Gamma, pc \vdash nch : \tau\ chan}$$

(INT_S) $$\Gamma, pc \vdash n : L\ val$$

(OP_S) $$\frac{\Gamma, pc \vdash e_1 : \tau_1\,\alpha, \qquad \Gamma, pc \vdash e_2 : \tau_2\,\alpha}{\Gamma, pc \vdash e_1\ \mathbf{op}\ e_2 : (\tau_1 \sqcup \tau_2)\ val}$$

(VAR_S) $$\frac{\Gamma(x) = \tau\,\alpha}{\Gamma, pc \vdash x : \tau\,\alpha}$$

(SKIP_S) $$\Gamma, pc \vdash\ \mathbf{skip}\ :\ H\ cmd$$

(ASSIGN -VAL_S) $$\frac{\Gamma, pc \vdash e : \tau\ val}{\Gamma, pc \vdash x := e : (\tau \sqcup pc)\ cmd, \Gamma \dagger [x \mapsto (\tau \sqcup pc)\ val]}$$

(ASSIGN -CHAN_S) $$\frac{\Gamma, pc \vdash e : \tau\ chan \qquad pc \preceq \tau}{\Gamma, pc \vdash x := e : \tau\ cmd, \Gamma \sqcup [\_instr \mapsto \tau \sqcap pc] \dagger [x \mapsto \tau\ chan]}$$

(RECEIVE-VAL_S) $$\frac{\Gamma(x_2) = \tau\ chan}{\Gamma, pc \vdash \mathbf{rec_c}\ x_1\ \mathbf{from}\ x_2 : (\tau \sqcup pc)\ cmd, \Gamma \dagger [x_1 \mapsto (\tau \sqcup pc)\ val]}$$

(RECEIVE-NAME_S) $$\frac{\Gamma(x_2) = \tau\ chan \qquad pc \preceq \tau}{\Gamma, pc \vdash \mathbf{rec_n}\, x_1\, \mathbf{from}\, x_2 : \tau\ cmd, \Gamma \sqcup [\_instr \mapsto \tau \sqcap pc] \dagger [x_1 \mapsto U\ chan]}$$

(SEND_S) $$\frac{\Gamma(x_1) = \tau_1\,\alpha, \qquad \Gamma(x_2) = \tau\ chan, \qquad (\tau_1 \sqcup pc) \preceq \tau}{\Gamma, pc \vdash \mathbf{send}\ x_1\ \mathbf{to}\ x_2 : \tau\ cmd, \overline{\Gamma}}$$

(COND_S) $$\frac{\Gamma, (pc \sqcup \tau_0) \vdash c_1 : \tau_1\ cmd, \Gamma' \qquad \Gamma, pc \vdash e : \tau_0\ val \qquad \Gamma, (pc \sqcup \tau_0) \vdash c_2 : \tau_2\ cmd, \Gamma'' \qquad \Gamma' \sqcup \Gamma'' \sqsupset \bot}{\Gamma, pc \vdash\ \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{end} : (\tau_1 \sqcap \tau_2)\ cmd, \Gamma' \sqcup \Gamma''}$$

(LOOP1_S) $$\frac{\Gamma, pc \vdash e : \tau_0\ val \qquad \Gamma, (pc \sqcup \tau_0) \vdash c : \tau\ cmd, \Gamma' \qquad \Gamma = \Gamma \sqcup \Gamma' \sqsupset \bot}{\Gamma, pc \vdash \mathbf{while}\ e\ \mathbf{do}\ c\ \mathbf{end} : \tau\ cmd\ \Gamma \sqcup \Gamma'}$$

(LOOP2_S) $$\frac{\Gamma, pc \vdash e : \tau_0\ val \quad \Gamma, (pc \sqcup \tau_0) \vdash c : \tau\ cmd, \Gamma' \quad \Gamma \neq \Gamma \sqcup \Gamma' \sqsupset \bot \qquad \Gamma \sqcup \Gamma', (pc \sqcup \tau_0) \vdash \mathbf{while}\, e\, \mathbf{do}\, c\, \mathbf{end} : \tau'\ cmd, \Gamma''}{\Gamma, pc \vdash \mathbf{while}\, e\ \mathbf{do}\, c\, \mathbf{end} : \tau'\ cmd,\ \Gamma''}$$

(SEQUENCE_S) $$\frac{\Gamma, pc \vdash c_1 : \tau_1\ cmd, \Gamma' \qquad \Gamma', pc \vdash c_2 : \tau_2\ cmd, \Gamma''}{\Gamma, pc \vdash c_1; c_2 : (\tau_1 \sqcap \tau_2)\ cmd, \Gamma''}$$

$\overline{\Gamma} = \Gamma \dagger [\_instr \mapsto U]$ whenever $(\tau_1 \sqcup pc) = \tau = U$ or $(\tau_1 \sqcup pc) \not\sqsubseteq \tau$.

Before explaining each rule in details, let us give more explanation on $\preceq$, which occurs in rule SEND_S, for example. As said earlier, this relation is used to ensure that rejection of a program will only occur if there is a flow from H to L; During type analysis, we distinguish between safe flow programs and

uncertain ones. For example, flows from U to H or L to U are secure because no matter what the types of uncertain variables actually are at runtime (L or H), the flows will always be secure. However, depending on the actual type of the U variable at runtime, a flow like U to L or U to U may be secure or not. A conservative analysis would reject a program with such flows but ours will tag the program as needing instrumentation and will carry on the type analysis. Consider the typing rule of the **send** instruction: the sending of x_1 on channel x_2 is accepted by the type system if $(\tau_1 \sqcup pc) \preceq \tau$, where τ_1 (resp. τ) is the type of x_1 (resp. x). This implies, by definition of $\preceq$, that if, for example, x_1 has an H content – or if the context is high – while x_2 is an L channel, then the rule cannot be applied, and consequently, the program will be rejected. Let us see how it works in other cases. Suppose that $(\tau_1 \sqcup pc) \preceq \tau$ and $(\tau_1 \sqcup pc) \sqsubseteq \tau$ but $\neg((\tau_1 \sqcup pc) = \tau = U)$; then the image of *_instr* under Γ is unchanged. This means that the flow from x_1 to x_2 is safe without any doubt. However, in all remaining cases, i.e., H to U, U to L and U to U, there is a risk of information leakage, and hence $\overline{\Gamma}(\_instr) = U$ indicating a need for instrumentation. The reason why we use a particular relation $\preceq$ instead of $\sqsubseteq$ is to allow the typing to progress until the end in case of uncertainty. If we had used the usual order relation $\sqsubseteq$, flows from H to U and U to L would have been rejected, even if there is a possibility that the variable of type U turns out to be a secure one. Handling this uncertainty is the main contribution of this paper and allows to reduce the number of false positive.

In related work, there are *subtyping judgements* of the form $\rho_1 \subseteq \rho_2$ or $\rho_1 \leq \rho_2$ [5,1]. For instance, given two security types τ and τ', if $\tau \subseteq \tau'$ then any data of type τ can be treated as data of type τ'. Similarly, if a command assigns contents only to variables of level H or higher then, *a fortiori*, it assigns only to variables L or higher; thus we would have $H\ cmd \subseteq L\ cmd$. In our work, we integrated those requirements directly in the typing rules. Instead of using type coercions, we assign a fixed type to the instruction according to the more general type. For two expressions e_1 and e_2 of type τ_1 and τ_2 respectively, e_1 **op** e_2 is typed $\tau_1 \sqcup \tau_2$. For two commands c and c' typed τ and τ', the composition through sequencing or conditionals is typed $\tau \sqcap \tau'$.

We now comment each of the typing rules. CHAN_S assigns types to channel constants, while VAR_S assigns types to variables. We assume that all integers have security type L and **skip** has type $H\ cmd$, the lowest security type of expressions and commands, respectively. OP_S assigns to the result the least upper bound of operands types.

ASSIGN-VAL_S and RECEIVE-VAL_S update the environment by mapping the modified variable to $(\tau \sqcup pc)\ val$. This way if we are in a block of instructions that need to be private (i.e., $pc = H$) then the modified variable will have type H. This is typically to prevent implicit flows in **while** and **if** statements when the condition is of type H.

ASSIGN-CHAN_S and RECEIVE-NAME_S both modify a channel variable; in the first case, the type of the channel is known, in the latter, it is not, and hence the variable is given type U. In both cases, the typing is done under the

```
if private                  if private                  if public
then x := lowValue          then c := publicChan        then c := publicChan
end                         else c := privanteChan      else c := privateChan end
send x to lowChan           end                         send highValue to c;
                            send lowValue to c;         send c to lowChan;
```

Fig. 1. Treating value and channel variables in different branches of the **if** construct

condition $pc \preceq \tau$ and generates instrumentation if $pc \sqcap \tau \sqsupseteq U$. Indeed, if the source (e in one case, x_2 in the other) is of type U or H, instrumentation must be performed to insert a test that will check if the actual type of x_1 is L: if it is the case, no send can be allowed on this channel, to avoid a downward flow (and hence the channel should be marked as unsecure in the dynamic analysis). In summary, there are three cases: if $pc \not\preceq \tau$, that is, $pc = H$ and $\tau = L$, the program is rejected; otherwise the program is accepted and instrumented except if $pc = L = \tau$. The case for assignation is illustrated by the second program of Fig. 1 (which uses a rule for **if** that we will explain later). In the last line, an information of low content is sent to c, but this cannot be allowed, as it would reveal information on our `private` condition: the sending cannot be blocked at that point; the flag must be lifted earlier, and this is when c is set to public while the context is high (because of the condition). We choose to reject such a clear flaw, which happens exactly when $pc \not\preceq \tau$ (see the end of this section for a sketch on how instrumentation will handle the case $pc = U$).

For the rule RECEIVE-VAL_S, x_1 is given the supremum of the type of channel x_2 and pc, since it receives the content of x_2 under the context type pc. Note that RECEIVE-VAL_S gives to the variable x_1 a $(\tau \sqcup pc)$ *val* type while RECEIVE-NAME_S assigns a U *chan* type.

For reasons explained earlier, SEND_S requires that the channel to which x_1 is sent must have a "maybe higher" (or equal) type than both x_1 and the context. Hence, the channel x_2, to which the content of x_1 is sent must satisfy $(\tau_1 \sqcup pc) \preceq \tau$; it assigns to the **send** instruction the type of channel x_2.

The rule COND_S requires to type the branches c_1 and c_2 under the type context $pc \sqcup \tau_0$. This prevents downward flows from the guard to the branches. A typical example of this situation is the first program of Fig. 1. Since x is typed under a high context (because of the `private` guard), it will obtain type H, and hence the program will be rejected, because x cannot be sent on the public channel. The typing environment produced, as stated by rule COND_S, is computed using the operator $\sqcup$. We now explain why $\sqcup$ is defined differently on channel variables and value variables. If Γ and Γ', the environments associated to the two branches of the **if** command, differ on a value variable, we choose to be pessimistic, and assign the supremum of the two security types. A user who prefers to obtain fewer false positive could assign type U to this variable, and leave the final decision to dynamic analysis. In the case of channel variables, we do not have the choice because the channel name can be private if it comes from a private source but its content can be public (and vice versa), as illustrated by the last program of Fig. 1. The last line should make the program rejected because

	Eval. & updates in 1st iteration	... in 2nd	... in 3rd
1. **receive**$_c$ h **from** `private`;	$h \mapsto H$		
2. $e, x_1, x_2, x_3 := 0$;	$e, x_1, x_2, x_3 \mapsto L\ val$		
3. **while** $e < 5$ **do**	$e < 5 : L\ val$		
4. **send** x_3 **to** `public`;	$[L \sqcup L \preceq L, pc \sqcup L = L], \_instr \mapsto L$	ok	ok
5. $x_3 := x_2$;	$x_3 \mapsto L\ val$	-	$x_3 \mapsto H\ val$
6. $x_2 := x_1$;	$x_2 \mapsto L\ val$	$x_2 \mapsto H\ val$	-
7. $x_1 := h$;	$x_1 \mapsto H\ val$	-	-
8. $e := e + 1$	-	-	-
9. **end**			

Fig. 2. The **while** construct needs iterative analysis

the **else** branch makes c a private information, hence we could conclude that the right typing for c when typing the **if** command is H but the penultimate line **send** `highValue` **to** c; would require that c be typed as L so that the program be rejected. Hence we must type c as U, justifying the definition of $\sqcup$.

SEQUENCE_S assigns to the command the greatest lower bound of the two sequential instructions types. Note that c_1 may update the typing environment from which c_2 is typed.

Our type analysis is flow-sensitive, thus we have to analyse the **while** construct in an iterative way. To illustrate this fact, consider the program on the left-hand side of Fig. 2. In this example, it is only on the fourth iteration that we can detect a security flaw, when x_3 finally enters the body of the while with a private content and when it is sent to a public channel. The corresponding typing computation is illustrated informally on the right-hand side of the figure. The need for iterative typing analysis is treated through two rules. The first rule, LOOP1_S, is the stopping condition: typing command c under Γ does not increase the type of any variable and hence no matter how many times c is repeated, the types of variables cannot be increased. Otherwise, in LOOP2_S, if the environment is modified, we iterate using the supremum of the produced environment and the original one. We claim that this process terminates because there is a finite number of variables in c, and because the operation $\sqcup$ between the original and the produced type environment is monotonic. Indeed, variables' types are monotonically modified with respect to $\sqsubseteq$ for value variable, and with respect to the following order for channel variables: $L \rightarrow H \rightarrow U$.

We now present an example illustrating the typing of a program that needs instrumentation. Consider the program of Fig. 3. The RECEIVE-NAME_S rule assigns to x the security type unknown $U\ chan$ since we are receiving a channel name; moreover, since we receive from a private channel, instrumentation is called (to block channel x if it happens to be of low type). Another call for instrumentation is illustrated in this example, as there is an explicit flow at instruction 4 from h, whose type is H, to x, whose type is unknown.

We conclude this section by sketching out how instrumentation will be implemented and discussing when false positive arise.

```
1. receive_c h from private;
2. if public then
3.    receive_n x from private;
4.    send h to x
5. end
```

Evaluations & updates
$h \mapsto H \;\; val$
$[pc \sqcup \tau_0 = L$ implies that branches context is $L]$
$[pc = L], \quad x \mapsto U \;\; chan, \;\; \_instr \mapsto U$
$[pc = L, \;\; H \sqcup pc \preceq U], \quad \_instr \mapsto U$

Fig. 3. A program generating unknown security values and calling for instrumentation

```
x := 0;                          c := highChannel
if public                        if private
then x := highValue end;         then c := lowChan
send x to lowChan.               end.
```

Fig. 4. False positives: uncertainty generated by **if** and assignation

False positives. Let us discuss the generation of false positives, that is, cases when we reject a program that is not potentially flawed. A reject can happen from the application of one of three rules: ASSIGN-CHAN_S , RECEIVE-NAME_S and SEND_S, when $pc \not\preceq \tau$, or $\tau_1 \sqcup pc \not\preceq \tau$. This is only possible for the pair of values (H, L), for which $H \not\preceq L$. If any of pc, τ_1 or τ is unknown (of type U), there will be no rejection, only instrumentation. According to our rules, type L can only be assigned if it is the true type of the variable, but H can be the result of a supremum taken in rule COND_S or LOOP_S. False positive can consequently occur from typing an **if** or **while** command whose guard prevent a bad branch to be taken; an example would be the first program of Fig. 4. If `public` is always false, the program is safe but SEND_S will reject it; this is because of the supremum taken in COND_S. A user who does not want false positive at all could change the settings in order that $H \;\; val \sqcup L \;\; val = U \;\; val$ instead of $H \;\; val$. The other situation where a false positive can occur is related to assignation and reception of a channel name. Consider the second program of Fig. 4. If `private` is always false or if c is never used later on in the program, this program is harmless but still rejected. Here again, there is a way to avoid the false positive, it would be to introduce a fourth security type, B, that would mark c as unsecure and would be tested by SEND_S – this is future work. In summary, the false positive cases that we detect happen in all other static analysis work, where they are considered reasonable, but we do suggest ways to circumvent them.

Instrumentation. When the type analysis concludes that some instrumentation of the program is needed, the program will be modified in two ways. Tests will be inserted before each problematic instruction, to check if it can be safely executed; to do so, we need to update, at runtime, the type of modified variables as well as the type of the context, and hence the program must be modified in consequence. Of course, we have access to the map *TypeOf_Channel*, defined a priori. We will store the variables types in a map, *TypeOf_Var*, the security type of the context in a stack, *Ctxt*. We push the type of a condition in *Ctxt* at the beginning of a conditional or a loop and we pop a type context after each end.

```
1.      receive_c h from private;
add_1.  Update(TypeOf_Var, h, (TypeOf_Channel(private) ⊔ top(Ctxt)) val);
2.      if public then
add_2.  push(L ⊔ top(Ctxt), Ctxt)                    ⟨because public : L  val⟩
3.        receive_n x from private;
add_3.    Update(TypeOf_Var, x, TypeOf_Channel(x));
          if TypeOf_Var(x) = L ∧ (TypeOf_Var(private) ⊔ top(Ctxt) = H)
          then unsecure(x) = true    else unsecure(x) = false;
add_4.  if (TypeOf_Var(h) ⋢ TypeOf_Channel(x)) ∨ unsecure(x) then alert
4.      else send h to x end
5.      end;
add_5.  pop(Ctxt)
```

Fig. 5. Projected instrumentation of the program of Fig. 3

In our type system there are three rules that can call for instrumentation; these "calls" happen when the variable *_instr* is assigned the value U or H. This occurs in the ASSIGN-CHAN_S rule, the RECEIVE-NAME_S rule and in the SEND_S. We will implement the type inference algorithm so as to uniquely identify statements either by labels or by the line number where they appear. The inference algorithm will save the identifier of the statement needing instrumentation, the types, variables, expressions and statements involved in the current statement. The instrumentation step will insert a test before each statement needing instrumentation. As an example, instrumenting the program of Fig. 3 will result into the one of Fig. 5. $Update(M, x, v)$ is a function that updates the map M with the association $x \mapsto v$, and *push*, *pop* and *top* are the usual functions on stacks. Note that the **receive$_\mathbf{n}$** instruction marks variable x as unsecure and this information is tested when we get to the **send** instruction on line `add_4`. Hence, if a public variable was received on a private channel on line `3`, then an alert would be sent – and some action must be taken to prevent the leak of information, like aborting the program or ignoring the sending. This sketch consists in a first approach. We think it is possible to make this instrumentation lighter if we use a data flow analysis. For instance, instead of inserting the test before a **send** we could insert it after a **receive$_\mathbf{n}$** statement, if we know that the channel will be used in a **send** and that it could yield an illegal flow. We plan to completely specify the instrumentation in our next step for this work.

4 Type System Soundness

In this section, we present the proof that our type system is sound, i.e., well-typed programs satisfy non-interference. Note that we only need to prove non-interference in case the program is typed without need of instrumentation, that is, when the type of *_instr* is L. If the type of *_instr* is U, non-interference will be guaranteed by instrumentation.

The soundness proof is inspired by the one sketched in [1]. We start by showing that our type system has the two properties listed earlier: *simple security* and *confinement*. We recall that α is to replace *val* or *chan*.

Lemma 1. *(Simple security) Given* $\Gamma, pc \vdash e : \tau\,\alpha$*, then for every variable* x *in* e*, such that* $\Gamma(x) = \tau'\,\alpha$*, we have* $\tau' \sqsubseteq \tau$*.*

The proof is by induction and is omitted for lack of space.

Lemma 2. *(Confinement) Let* Γ, Γ' *be typing environments,* pc *be a context type and* c *a command. Assume that* $\Gamma, pc \vdash c : \tau\ cmd, \Gamma'$ *with* $\Gamma'(\_instr) = L$*, then for every variable* x *modified in* c*, such that* $\Gamma'(x) = \tau'\,\alpha$*, we have* $\tau' \sqsupseteq \tau \sqcup pc$*. via the* **send** *command, we have* $\tau'' \sqsupseteq pc$*.*

We say that a variable x is modified in a command c if a value is assigned to x in c. The proof is omitted for lack of space. The following lemmas can be proved by usual induction on the structure of commands.

Lemma 3 ([1]). *Let* μ *and* μ' *be memories and* c *be a command. If* $\langle c, \mu\rangle \to \mu'$*,* $x \in dom(\mu)$ *and* x *is not modified in* c*, then* $\mu(x) = \mu'(x)$*.*

Non-interference essentially means that a variation of program input associated with a given security level does not cause variation of output of lower security level. This policy allows to manipulate and modify private data, as long as visible output does not improperly reveal information about that private data. Input data enter the program through the **receive** instructions, while output data are accessible through **send** instruction. Consequently, we define a relation $\sim_\tau$ on memories of programs that characterizes the observational power of an attacker on input and output data through communication channels.

Definition 1. *(Channel equality) Two channels* nch_1 *and* nch_2 *are equal, written* $nch_1 =_{ch} nch_2$ *if* $content(nch_1) = content(nch_2)$*, where* $content(ch)$ *is the sequence of data of channel* ch*.*

Intuitively, $nch_1 =_{ch} nch_2$ means that the channels contain exactly the same information; thus, two equal channels remain equal after the same update.

Definition 2. *(*τ*-equivalence) Two memories* μ *and* ν *are* τ*-equivalent, written* $\mu \sim_\tau \nu$*, if* $\forall x \in dom(\mu) \cap dom(\nu) : (\Gamma(x) = \tau'\ \ chan \wedge \tau' \sqsubseteq \tau) \Rightarrow \mu(x) =_{ch} \nu(x)$*.*

Since we consider only one observable class of security L, we use $\sim_L$ in the following, observing that this can be generalized to more classes. The relation $\sim_L$ associates two memories that are indistinguishable to an attacker who only has public (L) access privileges.

Even though our interest is on channels, because they are the only way of communicating with a program, we need to ensure that local integer variables of type L that are equally initialized in two memories L-equivalent remain equal through program execution. Indeed, the content of a local variable can be transferred to an observable channel. This is formalized by the following lemma:

Lemma 4 ([1]). *Given memories* $\mu \sim_L \nu$*, if for every integer variable* x *of type* L *val,* μ *and* ν *agree on the value of* x*, written* $\mu \sim_L^{val} \nu$*, then after program execution that produces the memories* μ' *and* ν'*, we have* $\mu' \sim_L^{val} \nu'$*.*

The proof is omitted; it is clear that variables can only be modified with a content coming from either equal channels (by $\sim_L$), or initially equal variables.

Non-interference is formalized as follows, a variant of the definition of [5]:

Definition 3. *(Non-interference) A program P satisfies non-interference if, for any memories $\mu \sim_L \nu$ such that $\mu \sim_L^{val} \nu$, the memories μ' and ν' produced by running P on μ and ν are also L-equivalent (if both runs terminate successfully).*

In this definition we consider that two runs of P would get exactly the same external processes writings on public channels. On the other hand, we do not treat the issue of the leak of information sensitive to termination. This issue is discussed in [8], thus we leave the non-interference definition as it was classically defined. Now let us state and prove the soundness theorem. The proof is omitted.

Theorem 1. *(Soundness theorem) Let P be a well-typed program under environment Γ, without need of instrumentation, and two L-equivalent memories μ and ν, i.e., $\mu \sim_L \nu$, that agree on integer variables of type L. If P runs successfully on both μ and ν producing the memories μ' and ν', then $\mu' \sim_L \nu'$.*

5 Related Work

Securing flow information has been widely studied since the late seventies. Denning and Denning [9] introduced secure information-flow by static analysis, based on control and data flow analysis. They defined implicit and explicit flow and devised static analysis and dynamic analysis based on lightly instrumenting the target program. Many static approaches have been devised based on type systems, they addressed languages with different levels of expressivity.

Smith [10] and Volpano and Smith [1] devised a type based analysis for an imperative language. Pottier and Simonet treat in [11] the functional language ML, supporting references, exceptions and polymorphism. In [12] Myers statically enforces information flow policy in JFlow, an extension of Java that adds level security annotations to variables, making information flow checking more precise and flexible. JFlow supports objects, subclassing, dynamic type tests, access control, and exceptions. Banerjee and Naumann devised a type based analysis in[13] that ensures secure flow information. They treat the object-oriented language Java. Barthe and al. in [14] and Terauchi and Aiken in [15] investigated logical-formulation of non-interference, enabling the use of theorem-proving or model-checking based techniques. In [16] Sabelfeld and Sands extend type system approaches to support higher order functions and non determinism. If the program is well typed according to the type system, then it satisfies the security property. We mention some studies on other notions of non-interference, possibilistic and probabilistic non-interference, which treat information flow security for concurrent programs, see [17,18,19,20].

Purely static approaches suffer from a large number of false positives, leading to reject programs that may be flow information secure. An analysis has been proposed to take into account data flow information [2], this approach is called

flow sensitive type approach. Combining static and dynamic approaches has been proposed by Russo and Sabelfeld in [3], where the authors prove that this approach reject less safe programs. Their approach is based on calling static analysis during execution.

Similarly to [2] and [3] our approach is flow sensitive, it differs in that we distinguish between variables in live memory and channels. We consider that programs interact with their environment through information flow via channels and that the security level is associated to an information according to its source, the channels from which this value is computed. We propose a three valued typing analysis aiming at the distinction between the cases where the analysis is uncertain due to lack of information and the cases where it is certain. Rather than calling static analysis during execution, our approach prepare for a light instrumentation of the code only when there is a need for it.

6 Conclusion

Ensuring secure information flow within sensitive systems has been studied extensively. In general, the key idea in type-based approaches is that if a program is well typed according to its typing rules, then it is secure according to given security properties.

We define a sound type system that captures lack of information in a program at compile-time. Our type system is flow-sensitive, variables are assigned the security levels of the values they store. We make a clear distinction between local variables and channels through which the program communicates. This makes our analysis more realistic.

Our main contribution is the handling of a three-valued security typing. The program is considered well typed, ill typed or uncertain. In the first case, the program can safely be executed, in the second case the program is rejected and need modifications, while in the third case instrumentation is to be used in order to guarantee the satisfaction of non-interference. This approach allows to eliminate false positives due to conservative static analysis approximations and to introduce run-time overhead only when it is necessary. We obtain fewer false positives than purely static approaches because we send some usually rejected programs to instrumentation.

Future work includes the extension of our type analysis to make it a complete security hybrid analysis, using data-flow analysis and code instrumentation. In short, we will insert tests before the instructions that were detected as possibly faulty during the type analysis.

Acknowledgements. The authors wish to thank the first referee whose arguments were really helpful and valuable in producing the final version of this paper.

References

1. Volpano, D., Irvine, C., Smith, G.: A sound type system for secure ow analysis. Journal of Computer Security 4(2-3), 167–187 (1996)
2. Hunt, S., Sands, D.: On flow-sensitive security types. In: Proceedings of the ACM Symposium on Principles of Programming Languages (January 2006)
3. Russo, A., Sabelfeld, A.: Dynamic vs. static flow-sensitive security analysis. In: Proceedings of the IEEE Computer Security Foundations Symposium (2010)
4. Denning, D.E.: A lattice model of secure information flow. Communications of the ACM 19, 236–243 (1976)
5. Smith, G.: Principles of secure information flow analysis. In: Malware Detection, vol. 27, pp. 291–307. Springer (2007)
6. O'Neill, K.R., Clarkson, M.R., Chong, S.: Information-flow security for interactive programs. In: Proceedings of the IEEE Computer Security Foundations Workshop (July 2006)
7. Kobayashi, N.: Type-based information flow analysis for the pi-calculus. Acta Informatica 42(4-5), 291–347 (2005)
8. Askarov, A., Hunt, S., Sabelfeld, A., Sands, D.: Termination-Insensitive Noninterference Leaks More Than Just a Bit. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 333–348. Springer, Heidelberg (2008)
9. Denning, D.E., Denning, P.J.: Certi cation of programs for secure information flow. Communications of the ACM 20, 504–513 (1977)
10. Smith, G.: A new type system for secure information flow. In: Proceedings of the IEEE Workshop on Computer Security Foundations, pp. 115–125 (2001)
11. Pottier, F., Simonet, V.: Information flow inference for ML. ACM Transactions on Programming Languages and Systems 25, 117–158 (2003)
12. Myers, A.C.: J ow: Practical mostly-static information flow control. In: Proceedings of the ACM Symposium on Principles of Programming Languages (1999)
13. Banerjee, A., Naumann, D.A.: Secure information flow and pointer con nement in a java-like language. In: Proceedings of the IEEE Computer Security Foundations Workshop (2002)
14. Barthe, G., D'Argenio, P.R., Rezk, T.: Secure information flow by self-composition. In: Proceedings of the IEEE Workshop on Computer Security Foundations (2004)
15. Terauchi, T., Aiken, A.: Secure Information Flow as a Safety Problem. In: Hankin, C., Siveroni, I. (eds.) SAS 2005. LNCS, vol. 3672, pp. 352–367. Springer, Heidelberg (2005)
16. Sabelfeld, A., Sands, D.: A per model of secure information flow in sequential programs. Higher-Order and Symbolic Computation 14(1), 59–91 (2001)
17. Barthe, G., Prensa Nieto, L.: Secure information flow for a concurrent language with scheduling. Journal of Computer Security 15, 647–689 (2007)
18. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: Proceedings of the IEEE Workshop on Computer Security Foundations (2000)
19. Smith, G.: Probabilistic noninterference through weak probabilistic bisimulation. In: Proceedings of the IEEE Computer Security Foundations Workshop (June-July 2003)
20. Smith, G.: Improved typings for probabilistic noninterference in a multi-threaded language. Journal of Computer Security 14(6), 591–623 (2006)

Towards the Orchestration of Secured Services under Non-disclosure Policies*

Tigran Avanesov[1,2,3], Yannick Chevalier[2],
Michaël Rusinowitch[1], and Mathieu Turuani[1]

[1] INRIA Nancy Grand Est,
615 allée du jardin botanique, 54000 Vandœuvre-lés-Nancy, France
{rusi,turuani}@inria.fr
[2] IRIT, Université de Toulouse,
118 route de Narbonne, F-31062 Toulouse, France
ychevali@irit.fr
[3] SnT, Université du Luxebmourg,
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
tigran.avanesov@uni.lu

Abstract. The problem of finding a mediator to compose secured services has been reduced in our former work to the problem of solving deducibility constraints similar to those employed for cryptographic protocol analysis. We extend in this paper the mediator synthesis procedure by a construction for expressing that some data is not accessible to the mediator. Then we give a decision procedure for verifying that a mediator satisfying this non-disclosure policy can be effectively synthesized. This procedure has been implemented in CL-AtSe, our protocol analysis tool. The procedure extends constraint solving for cryptographic protocol analysis in a significative way as it is able to handle negative deducibility constraints without restriction. In particular it applies to all subterm convergent theories and therefore covers several interesting theories in formal security analysis including encryption, hashing, signature and pairing.

Keywords: Web services, Orchestration, security policy, separation of duty, deducibility constraints, cryptographic protocols.

1 Introduction

1.1 Context

Trust and security management in distributed frameworks is known to be a non-trivial critical issue. It is particularly challenging in Service Oriented Architecture where services can be discovered and composed in a dynamic way. Implemented solutions should meet the seemingly antinomic goals of openness and flexibility on one hand and compliance with data privacy and other regulations on the other hand. We have demonstrated in previous works [7,25,2]

* This work is supported by FP7 AVANTSSAR [5] and FP7 NESSoS [22] projects.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 130–145, 2012.

that functional agility can be achieved for services with a message-level security policy by providing an automated service synthesis algorithm. It resolves a system of deducibility constraints by synthesizing a *mediator* that may adapt, compose and analyze messages exchanged between client services and having the functionalities specified by a goal service. It is complete as long as the security policies only apply to the participants in the orchestration. But once the policies also concern the synthesized service or the eligibility of the communication participants, the cited method losses the completenes as an approach to sove the problem. However security policies often deals with such kind of requirements. For instance an organisation may not be trusted to efficiently protect the customer's data against attackers even though it is well-meaning. In this case a client would require that the mediator synthesized to interact with this organization must not have direct access to her private data, which is an effective protection even in case of total compromise. Also it is not possible to specify that the mediator enforces *e.g.* dynamic separation of duty, *i.e.*, restrictions on the possible participants at some step. The *non-deducibility* constraints help to express such types of policies.

Since checking whether a solution computed by our previous algorithm satisfies the non-deducibility constraints is not complete, we propose in this paper to solve during the automated synthesis of the mediator both deducibility and non-deducibility constraints. The former are employed to specify a mediator that satisfies the functional requirements and the security policy on the messages exchanged by the participants whereas the latter are employed to enforce a security policy on the mediator and the participants to the orchestration.

Original contribution. We have previously proposed decision procedures [7,25,2] for generating a mediator from a high-level specification with deducibility constraints of a goal service. In this paper we extend the formalism to include non-deducibility constraints in the specification of the mediator and provide a decision procedure synthesizing a mediator for the resulting constraint systems.

Related works. In order to understand and anticipate potential flaws in complex composition scenarios, several approaches have been proposed for the formal specification and analysis of secure services [11,9]. Among the works dedicated to trust in multi-agent systems, the models closest to ours are [13,16] in which one can express that an agent trusts another agent in doing or forbearing of doing an action that leads to some goal. To our knowledge no work has previously considered the automatic orchestration of security services with policies altogether as ours. However there are some interesting related attempts to analyze security protocols and trust management [18,12]. In [18] the author uniformly models security protocols and access control based on trust management. The work introduces an elegant approach to model automated trust negotiation. We also consider an integrated framework for protocols and policies but in our case *i*) policies can be explicitly negative such as non-disclosure policies and separation-of-duty *ii*) we propose a decision procedure for the related trust negotiation problem *iii*) we do not consider indistinguishability properties. In [12]

security protocols are combined with authorization logics that can be expressed with acyclic Horn clauses. The authors encode the derivation of authorization predicates (for a service) as subprotocols and can reuse in that way the constraint solving algorithm from [20] to obtain a decision procedure. In our case we consider more general intruder theories (subterm convergent ones) but focus on negation. We conjecture that our approach applies to their authorization policies too.

Our decision procedure for general (negative and positive) constraints extends [8] where negative constraints are limited to have ground terms in right-hand sides, and the deduction system is Dolev-Yao system [10], a special instance of the subterm deduction systems we consider here. In [15] the authors study a class of contract signing protocols where some very specific Dolev-Yao negative constraints are implicitly handled.

Finally one should note that the non deducibility constraints we consider tell that some data cannot be disclosed *globally* but they cannot express finer-grained privacy or information leakage notions relying on probability such as for instance differential privacy.

Paper organization. In Subsection 1.2 we introduce a motivating banking application and sketch our approach to obtain a mediator service. To our knowledge this application is out of the scope of alternative automatic methods. In Section 2 we present our formal setting. A deduction system (Subsection 2.2) describes the abilities of the mediator to process the messages. The mediator synthesis problem is reduced to the resolution of constraints that are defined in Subsection 2. In Section 3 we recall the class of *subterm deduction systems* and their properties. These systems have nice properties that allow us to decide in Section 4 the satisfiability of deducibility constraints even with negation. Finally we conclude in Section 5.

1.2 Synthesis of a Loan Origination Process (LOP)

We illustrate how negative constraints are needed to express elaborated policies such as Separation of Duty by a classical loan origination process example. Our goal is to synthesize a mediator that selects two bank clerks satisfying the Separation of Duty policy to manage the client request. Such a problem is solved automatically by the decision procedure proved in the following sections. Let us walk through the specification of the different parts of the orchestration problem.

Formal setting. Data are represented by first-order terms defined on a signature that comprises binary symbols for symmetric and assymetric encryptions (resp. $\{|\_|\}\_$, $\{\_\}\_$), signature ($\{\_\}^{\mathrm{sig}}_{\_}$), and pairing (pair). Given a public key k we write $\mathrm{inv}(k)$ its associated private key. For example $\{a\}^{\mathrm{sig}}_{\mathrm{inv}(k)}$ is the signature of a by the owner of public key k. For readability we write $a.b.c$ a term pair $(a, \mathrm{pair}(b, c))$. The binary symbol rel expresses that two agents are related and is used for defining a Separation of Duty policy. A unary symbol g is employed to designate participants identity in the "relatives" database.

Client and clerks. The client and the clerks are specified by services with a security policy, specifying the cryptographic protections and the data and security tokens, and a business logic that specify the sequence in which the operations may be invoked. These are compiled into a sequence of protected messages (depicted in Fig. 1 and 2 and explained in the following paragraphs) each service is willing to follow during the orchestration.

Client C wants to ask for a loan from a service P, but for this he needs to get an approval from two banking clerks. He declares his intention by sending to mediator M a signed by him message containing service name P and the identity of the client $g(C)$. The mediator should send back the names of two clerks A and B who will evaluate his request. The client then sends to each clerk a request containing amount $Amnt$, his name C and a fresh key N_k which should be used to encrypt decisions. Each request is encrypted with a public key of the corresponding clerk ($pk(A)$ or $pk(B)$). Then the mediator must furnish the decisions (R_a and R_b) of two clerks each encrypted with the proposed key N_k and also their signatures. Finally, the client uses these tokens to ask his loan from P, where $pk(P)$ is a public key of P.

Clerk A receives a request to participate in a LOP which is conducted by mediator M. If he accepts, he returns his identity and public key. Then Clerk receives the client's request for a loan to evaluate: amount Amnt, client's name C and a temporary key K for encrypting his decision. The last is sent back together with a signature certifying the authenticity of this decision on the given request.

The client's non-disclosure policy is given in Fig. 2 and is self-explanatory: the mediator should not know the amount of the loan and should not be able to know the decisions of clerks.

Let us explain the services' non-disclosure policy. The Clerk's decision (its last message) should be unforgeable, thus, it should not be known by the Mediator before it was sent by the Clerk (first non-disclosure constraint of Fig. 1). The role clerk played by A can be used by the mediator only if the constraint $\natural g(A)$ is satisfied, showing that A is not a relative with any other actor of the protocol, as client and the other clerk (second non-disclosure constraint of Fig. 1).

Goal service. In contrast with the other services and clients, the goal service is only described in terms of possible operations and available initial data.

Initial data. Beside his private/public keys and the public keys of potential partners (*e.g.* $\mathrm{pk}(P)$) the goal service has access to a relational database $\mathrm{rel}(g(a), g(c)), \mathrm{rel}(g(b), g(c)), \ldots$ for storing known existing relations between agents to be checked against conflict of interests.

Deduction rules. The access to the database as well as the possible operations on messages are modeled by a set of deduction rules (formally defined later). We anticipate on the rest of this paper, and present the rules specific to this case study grouped into composition and decomposition rules in Fig. 3. In fact, these rules represent Dolev-Yao deduction system for symmetric/assymetric encryption, signature and pairing augmented with two rules for querying rel database.

Clerk's (A) communications:[1]

$* \Rightarrow \mathrm{A} : \mathrm{request.M}$
$\mathrm{A} \Rightarrow \mathrm{M} : g(A).\,\mathrm{pk}\,(A)$
$\mathrm{M} \Rightarrow \mathrm{A} : \{Amnt.C.K\}_{\mathrm{pk}(A)}$
$\mathrm{A} \Rightarrow \mathrm{M} : m_1(A, Resp_A, K, C, Amnt)$

Non-disclosure constraints:

1. M cannot deduce the last message before it is sent by A.
2. M cannot deduce $g(A)$ before the second message is sent by A.

Fig. 1. Clerk's communications and non-disclosure constraints

Client's (C) communications:[1]

$\mathrm{C} \Rightarrow \mathrm{M} : \{g(C).loan.P\}^{\mathrm{sig}}_{\mathrm{inv}(pk(C))}$
$\mathrm{M} \Rightarrow \mathrm{C} : A.B$
$\mathrm{C} \Rightarrow \mathrm{M} : m_2(A, Amnt).m_2(B, Amnt)$
$\mathrm{M} \Rightarrow \mathrm{C} : m_3(A, R_a).m_3(B, R_b)$
$\mathrm{C} \Rightarrow \mathrm{P} : m_4(\mathrm{pk}\,(P)\,, A, B, R_a, R_b)$

Non-disclosure constraints:

1. M cannot deduce the amount $Amnt$.
2. M cannot deduce A's decision R_a.
3. M cannot deduce B's decision R_b.

Fig. 2. Client's Communications and non-disclosure constraints

Composition rules	Decomposition rules	
$x, y \to \mathrm{pair}\,(x, y)$	$\mathrm{pair}\,(x, y) \to x$	
	$\mathrm{pair}\,(x, y) \to y$	$x, \mathrm{rel}(x, y) \to y$
$x, y \to \{\lvert x\rvert\}_y$	$y, \{\lvert x\rvert\}_y \to x$	$y, \mathrm{rel}(x, y) \to x$
$x, y \to \{x\}_y$	$\mathrm{inv}\,(y)\,, \{x\}_y \to x$	
$x, \mathrm{inv}\,(y) \to \{x\}^{\mathrm{sig}}_{\mathrm{inv}(y)}$	$y, \{x\}^{\mathrm{sig}}_{\mathrm{inv}(y)} \to x$	

Fig. 3. Deduction system for the LOP example

Mediator synthesis problem. In order to communicate with the services (here the client, the clerks and the service P), a mediator has to satisfy a sequence of constraints expressing that *(i)* each message m expected by a service (denoted $?m$) can be deduced from all the previously sent messages m' (denoted $!m'$) and the initial knowledge and *(ii)* each message w that should not be known or disclosed (denoted $\natural w$ and called negative constraint) is not deducible.

The orchestration problem consists in finding a satisfying interleaving of the constraints imposed by each service. For instance, clerk's and client's constraints extracted from Fig. 1 and Fig. 2 are:

$$\begin{cases} Client(C) \triangleq\ !_M \{g(C).loan.P\}^{\mathrm{sig}}_{\mathrm{inv}(K_C)}\ ?_M A.B\ !_M m_2(A, Amnt).m_2(B, Amnt) \\ \qquad\qquad ?_M m_3(A, R_a).m_3(B, R_b)\ \natural_M Amnt\ \natural_M R_A\ \natural_M R_B \\ \qquad\qquad !_P m_4(\mathrm{pk}\,(P)\,, A, B, R_a, R_b) \\ Clerk(A) \triangleq\ ?request.M\ \natural_M g(A)\ !_M g(A).\,\mathrm{pk}\,(A)\ ?_M \{Amnt.C.K\}_{\mathrm{pk}(A)} \\ \qquad\qquad \natural_M m_1(A, Resp_A, K, C, Amnt)\ !_M m_1(A, Resp_A, K, C, Amnt) \end{cases}$$

If it exists our procedure outputs a solution which can be translated automatically into a mediator. Note, for example, that without the negative constraint $\natural g(A)$ a synthesized mediator might accept any clerk identity and that could violate the Separation of Duty policy.

2 Derivations and Constraint Systems

In our setting messages are terms generated or obtained according to some elementary rules called *deduction rules*. A *derivation* is a sequence of deduction rules applied by a mediator to build new messages. The goal of the synthesis is specified by a *constraint system*, *i.e.* a sequence of terms labelled by symbols !,? or $\natural$, respectively *sent*, *received*, or *unknown* at some step of the process.

2.1 Terms and Substitutions

Let $\mathcal{X}$ be a set of *variables*, $\mathcal{F}$ be a set of *function symbols* and $\mathcal{C}$ a set of *constants*. The set of *terms* $\mathcal{T}$ is the minimal set containing $\mathcal{X}$, $\mathcal{C}$ and if $t_1, \ldots, t_k \in \mathcal{T}$ then $f(t_1, \ldots, t_k) \in \mathcal{T}$ for any $f \in \mathcal{F}$ with arity k. The set of *subterms* of a term t is denoted $\mathrm{Sub}(t)$ and is the minimal set containing t such that $f(t_1, \ldots, t_n) \in \mathrm{Sub}(t)$ implies $t_1, \ldots, t_n \in \mathrm{Sub}(t)$ for $f \in \mathcal{F}$. We denote $\mathrm{Var}(t)$ the set $\mathcal{X} \cap \mathrm{Sub}(t)$. A term t is *ground* is $\mathrm{Var}(t) = \emptyset$. We denote $\mathcal{T}_g$ the set of ground terms.

A *substitution* σ is an idempotent mapping from $\mathcal{X}$ to $\mathcal{T}$. It is ground if it is a mapping from $\mathcal{X}$ to $\mathcal{T}_g$. The application of a substitution σ on a term t is denoted $t\sigma$ and is equal to the term t where all variables x have been replaced by the term $x\sigma$. We say that a substitution σ is *injective* on a set of terms T, iff for all $p, q \in T$ $p\sigma = q\sigma$ implies $p = q$. The *domain* of σ (denoted by $\mathrm{dom}(\sigma)$) is set: $\{x \in \mathcal{X} : x\sigma \neq x\}$. The *image* of σ is $\mathrm{img}(\sigma) = \{x\sigma : x \in \mathrm{dom}(\sigma)\}$. Given two substitutions σ, δ, the substitution $\sigma\delta$ has for domain $\mathrm{dom}(\sigma) \cup \mathrm{dom}(\delta)$ and is defined by $x\sigma\delta = (x\sigma)\delta$. If $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\delta) = \emptyset$ we write $\sigma \cup \delta$ instead of $\sigma\delta$.

A *unification system* U is a finite set of equations $\{p_i =_? q_i\}_{1 \leq i \leq n}$ where $p_i, q_i \in \mathcal{T}$. A substitution σ is an *unifier* of U or equivalently satisfies U iff for all $i = 1, \ldots, n$, $p_i\sigma = q_i\sigma$. Any satisfiable unification system U admits a *most general unifier* $\mathrm{mgu}(U)$, unique modulo variable renaming, and such that for any unifier σ of U there exists a substitution τ such that $\sigma = \mathrm{mgu}(U)\tau$. Wlog we assume in the rest of this paper that $\mathrm{Var}(\mathrm{img}(\mathrm{mgu}(U))) \subseteq \mathrm{Var}(U)$, *i.e.*, the most general unifier does not introduce new variables.

A *sequence* s is indexed by $[1, \ldots, n]$ with $n \in \mathbb{N}$. We write $|s|$ the length of s, $\emptyset$ the empty sequence, $s[i]$ the ith element of s, $s[m:n]$ the sequence $s[m], \ldots, s[n]$ and s, s' the concatenation of two sequences s and s'. We write $e \in s$ and $E \subseteq s$ for, respectively, $\exists i : s[i] = e$ and $\forall e \in E, e \in s$.

2.2 Deduction Systems

The new values created by the mediator are constants in a subset $\mathcal{C}_{\mathrm{med}}$ of $\mathcal{C}$. We assume that both $\mathcal{C}_{\mathrm{med}}$ and $\mathcal{C} \setminus \mathcal{C}_{\mathrm{med}}$ are infinite. Given $l_1, \ldots, l_n, r \in \mathcal{T}$,

[1] We have employed the following abbreviations for messages:

$$\begin{cases} m_1(A, Resp, K, Ct, S) = \{h(A.S.Ct.Resp)\}^{\mathrm{sig}}_{\mathrm{inv}(pk(A))}.\{|Resp|\}_K \\ m_2(A, S) = \{S.C.N_k\}_{\mathrm{pk}(A)} \\ m_3(A, R) = m_1(A, R, N_k, C, Amnt) \\ m_4(K_0, A, B, R_1, R_2) = \{Amnt.C.A.R_1.B.R_2\}_{K_0}.m_3(A, R_1).m_3(B, R_2) \end{cases}$$

the notation $l_1, \ldots, l_n \to r$ denotes a *deduction rule* if $\mathrm{Var}(r) \subseteq \bigcup_{i=1}^{n} \mathrm{Var}(l_i)$. A *deduction* is a ground instance of a deduction rule. A *deduction system* is a set of deduction rules that contains a finite set of deduction rules in addition to all *nonce creation rules* $\to n$ (one for every $n \in \mathcal{C}_{\mathrm{med}}$) and all reception rules $?t$ (one for every $t \in \mathcal{T}$). All rules but the reception rules are called *standard* rules. The deduction system describes the abilities of the mediator to process the messages. In the rest of this section we fix an arbitrary deduction system $\mathcal{D}$. We denote by $l \twoheadrightarrow r$ any rule and $l \to r$ any standard rule.

2.3 Derivations and Localizations

A *derivation* is a sequence of deductions, including receptions of messages from available services, performed by the mediator. Given a sequence of deductions $E = (l_i \twoheadrightarrow r_i)_{i=1,\ldots,m}$ we denote $\mathrm{R}_E(i)$ the set $\{r_j : j \leq i\}$.

Definition 1 (Derivation). *A sequence of deductions $D = (l_i \twoheadrightarrow r_i)_{i=1,\ldots,m}$ is a* derivation *if for any $i \in \{1, \ldots, m\}$, $l_i \subseteq \mathrm{R}_D(i-1)$.*

Given a derivation D we define $\mathrm{Next}_D(i) = \min(\{|D|+1\} \cup \{j : j > i \text{ and } D[j] = ?t_j\})$. The explicit knowledge of the mediator is the set of terms it has already deduced, and its implicit knowledge is the set of terms it can deduce. If the former is K we denote the latter $\mathrm{Der}(K)$. A derivation D is a *proof* of $s \in \mathrm{Der}(K)$ if $?r \in D$ implies $r \in K$, and $D[|D|] = l \twoheadrightarrow t$. Thus, we have:

$$\mathrm{Der}(K) = \{t : \exists D \text{ derivation s.t. } ?r \in D \text{ implies } r \in K, \text{ and } D[|D|] = l \to t\}$$

2.4 Constraint Systems

Definition 2 (Constraint system). *A constraint system $\mathcal{S}$ is a sequence of constraints where each constraint has one of three forms (where t is a term):*

1. *$?t$, denoting a message reception by an available service or a client,*
2. *$!t$, denoting a message emission by an available service or a client,*
3. *$\natural t$, a negative constraint, denoting that the mediator must not be able to deduce t at this point;*

and that satisfies the following properties for any $1 \leq i \leq |\mathcal{S}|$:

Origination: *if $\mathcal{S}[i] = !t_i$ then $\mathrm{Var}(t_i) \subseteq \bigcup_{j<i} \mathrm{Var}(\{t_j : \mathcal{S}[j] = ?t_j\})$;*
Determination: *if $\mathcal{S}[i] = \natural t_i$ then $\mathrm{Var}(t_i) \subseteq \bigcup_j \mathrm{Var}(\{t_j : \mathcal{S}[j] = ?t_j\})$.*

Origination means that every unknown in a service's state originates from previous input by the mediator. *Determination* means that negative constraints are on messages determined by a service's state at the end of its execution.

In the rest of this paper $\mathcal{S}$ (and decorations thereof) denotes a constraint system. An index i is a *send* (*resp.* a *receive*) index if $\mathcal{S}[i] = !t$ (*resp.* $\mathcal{S}[i] = ?t$) for some term t. If $i_1, \ldots, i_k$ is the sequence of all send (*resp.* receive) indices in $\mathcal{S}$ we denote $\mathrm{Out}(\mathcal{S})$ (*resp.* $\mathrm{In}(\mathcal{S})$) the sequence $\mathcal{S}[i_1], \ldots, \mathcal{S}[i_k]$. We note that the origination and determination properties imply $\mathrm{Var}(\mathcal{S}) = \mathrm{Var}(\mathrm{In}(\mathcal{S}))$. Given $1 \leq i \leq |\mathcal{S}|$ we denote $\mathrm{prev}_{\mathcal{S}}(i)$ to be $\max(\{0\} \cup \{j : j \leq i \text{ and } \mathcal{S}[j] = !t_j\})$.

Definition 3 (Solution of a constraint system). *A ground substitution σ is a solution of $\mathcal{S}$, and we denote $\sigma \models \mathcal{S}$, if* $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$ *and*

1. *if* $\mathcal{S}[i] = ?t$ *then* $t\sigma \in \mathrm{Der}(\{t_j\sigma : j \leq \mathrm{prev}_{\mathcal{S}}(i)$ *and* $\mathcal{S}[j] = !t_j\})$
2. *if* $\mathcal{S}[i] = \natural t$ *then* $t\sigma \notin \mathrm{Der}(\{t_j\sigma : j \leq \mathrm{prev}_{\mathcal{S}}(i)$ *and* $\mathcal{S}[j] = !t_j\})$

Definition 4 (Compliant derivations). *Let σ be a ground substitution with* $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$. *A derivation D is* $(\mathcal{S}, \sigma)$-compliant *if there exists a strictly increasing bijective mapping α from the send indices of $\mathcal{S}$ to the set $\{j : D[j] = ?r\}$ such that $\mathcal{S}[i] = !t$ implies $D[\alpha(i)] = ?t\sigma$.*

An example of $(\mathcal{S}, \sigma)$-compliant derivation is shown in Figure 4. Since a sequence of receptions is a derivation, we note that for every ground substitution σ with $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathrm{In}(\mathcal{S}))$ there exists at least one compliant derivation D.

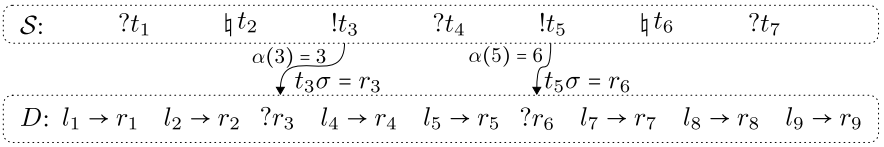

Fig. 4. A constraint system and a compliant derivation

Definition 5 (Proof of a solution). *Let σ be a ground substitution. A derivation D is a* proof *of $\sigma \models \mathcal{S}$, and we denote $D, \sigma, \alpha \vdash \mathcal{S}$, if:*

1. *D is $(\mathcal{S}, \sigma)$-compliant with the mapping α and*
2. *if $\mathcal{S}[i] = ?t$ there is $j < \mathrm{Next}_D(\alpha(\mathrm{prev}_{\mathcal{S}}(i)))$ such that $D[i] = l \twoheadrightarrow t\sigma$ and*
3. *if $\mathcal{S}[i] = \natural t$ then $t\sigma \notin \mathrm{Der}(\{t_j\sigma : j \leq \mathrm{prev}_{\mathcal{S}}(i)$ and $\mathcal{S}[j] = !t_j\})$.*

In Figure 4, if σ is a solution of $\mathcal{S}$ and, for example, $t_1\sigma = r_2$, $t_2\sigma \notin \mathrm{Der}(\emptyset)$, $t_4\sigma = r_4$, $t_6\sigma \notin \mathrm{Der}(\{r_3, r_6\})$ and $t_7\sigma = r_8$ then D is a proof of $\sigma \models \mathcal{S}$.

Let us prove that if $\sigma \models \mathcal{S}$ then there is a proof $D, \sigma, \alpha \vdash \mathcal{S}$.

Definition 6 (Maximal derivation). *Let T be a finite set of terms and σ be a ground substitution with* $\mathrm{dom}(\sigma) = \mathrm{Var}(T)$. *A derivation D is (T, σ)-maximal iff for every $t \in \mathrm{Sub}(T)$, $t\sigma \in \mathrm{Der}(\mathrm{R}_D(i))$ implies $t\sigma \in \mathrm{R}_D(\mathrm{Next}_D(i) - 1)$.*

First we prove that maximal derivations are natural proof candidates of $\sigma \models \mathcal{S}$.

Lemma 1. *Let σ be a ground substitution with* $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$ *and D be a $(\mathcal{S}, \sigma)$-compliant $(\mathrm{Sub}(\mathcal{S}), \sigma)$-maximal derivation.* ***Then*** *$\sigma \models \mathcal{S}$ iff for all i*

- *if $\mathcal{S}[i] = ?t$ then there exists $j < \mathrm{Next}_D(\alpha(\mathrm{prev}_{\mathcal{S}}(i))) : D[j] = l \twoheadrightarrow t\sigma$ and*
- *if $\mathcal{S}[i] = \natural t$ then for all $j < \mathrm{Next}_D(\alpha(\mathrm{prev}_{\mathcal{S}}(i))) : D[j] \neq l \twoheadrightarrow t\sigma$.*

In the next lemma we show that any (T, σ)-maximal derivation D may be extended into a (T', σ')-maximal derivation for an arbitrary extension T', σ' of T, σ by adding into D only standard deductions.

Lemma 2. *Let σ be a ground substitution with $\mathrm{dom}(\sigma) = \mathrm{Var}(\mathcal{S})$. Let T_1, T_2 be two sets of terms such that $T_1 \subseteq T_2$, and σ_1, σ_2 be two substitutions such that $\mathrm{dom}(\sigma_1) = \mathrm{Var}(T_1)$ and $\mathrm{dom}(\sigma_2) = \mathrm{Var}(T_2) \setminus \mathrm{Var}(T_1)$. If D is a (T_1, σ_1)-maximal $(\mathcal{S}, \sigma)$-compliant derivation in which no term is deduced twice by a standard rule,* ***then*** *there exists a $(T_2, \sigma_1 \cup \sigma_2)$-maximal $(\mathcal{S}, \sigma)$-compliant derivation D' in which no term is deduced twice by a standard rule such that every deduction whose right-hand side is in $\mathrm{Sub}(T_1)\sigma_1$ occurs in D' iff it occurs in D.*

Proof. Let $i_1, \ldots, i_k$ be the indices of the non-standard rules in D, let $D[i_j] = ?t_{i_j}$, and let for $0 \leq j \leq k$ $D_j = D[i_j{+}1 : i_{j+1}{-}1]$ with $i_0 = 0$ and $i_{k+1} = |D|{+}1$. That is, $D = D_0, !t_{i_1}, D_1, !t_{i_2}, D_2, \ldots !t_{i_k}, D_k$. Noting that $\mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2) = \emptyset$ let $\sigma' = \sigma_1 \cup \sigma_2$.

For each $t \in \mathrm{Sub}(T_2)$ such that $t\sigma' \in \mathrm{Der}(t_{i_1}, \ldots, t_{i_k})$ let i_t be minimal such that $t\sigma' \in \mathrm{Der}(t_{i_1}, \ldots, t_{i_t})$, and let E_t^0 be a proof of this fact, and E_t be a sequence of standard deductions obtained by removing every non-standard deduction from E_t^0.

For $0 \leq j \leq k$ let D'_j be the sequence of standard deduction steps $D_j, E_{s_1}, \ldots, E_{s_p}$ for all $s_m \in \mathrm{Sub}(T_2)\sigma' \setminus \mathrm{Sub}(T_1)\sigma'$ such that $i_{s_m} = j$ in which every rule of $E_{s_1}, \ldots, E_{s_p}$ that deduces a term previously deduced in the sequence or for some $m \leq j$ deduced in D'_m or in $D[i_m]$ is removed.

Let $D' = D'_0, ?t_{i_1}, D'_1, \ldots, ?t_{i_k}, D'_k$. We have deleted in each E_t^0 only deductions whose right-hand side occurs before in D', and thus D' is a derivation. Since the D'_i contains only standard deductions, we can see that D' is $(\mathcal{S}, \sigma)$-compliant.

Since D is (T_1, σ_1)-maximal and no term is deduced twice in D we note that, for $t \in T_1$, no standard deduction of $t\sigma_1$ from a sequence D_j is deleted. Furthermore we note that standard deductions of terms $T_2\sigma_2$ that are also in $T_1\sigma_1$ are deleted by construction and by the maximality of D. Thus a deduction whose right-hand side is in $\mathrm{Sub}(T_1)\sigma_1$ is in D' iff it occurs in D.

By construction D' is (T_2, σ')-maximal and no term is deduced twice by standard deductions.

Taking $T_1 = \emptyset$, $T_2 = \mathrm{Sub}(\mathcal{S})$, and $\sigma_2 = \sigma$, Lemma 2 implies that for every substitution σ of domain $\mathrm{Var}(\mathcal{S})$ there exists a $(\mathcal{S}, \sigma)$-compliant $(\mathrm{Sub}(\mathcal{S}), \sigma)$-maximal derivation D. By Lemma 1 if $\sigma \models \mathcal{S}$ then D is a proof of $\sigma \models \mathcal{S}$. Since the converse is trivial, it suffices to search proofs maximal wrt $T \supseteq \mathrm{Sub}(\mathcal{S})$.

3 Subterm Deduction System

3.1 Definition and Main Property

We say that a deduction system is a *subterm deduction system* whenever each deduction rule which is not a nonce creation or a message reception is either:

1. $x_1, \ldots, x_n \to f(x_1, \ldots, x_n)$ for a function symbol f;
2. $l_1, \ldots, l_n \to r$ for some terms $l_1, \ldots, l_n, r$ such that $r \in \bigcup_{i=1}^{n} \mathrm{Sub}(l_i)$.

A *composition* rule is either a message reception, a nonce creation, or a rule of the first type. A deduction rule is otherwise a *decomposition* rule. Reachability problemsfor deduction systems with a convergent equational theory are reducible to the satisfiability of a constraint system in the empty theory for a deduction system in our setting [17,14]. If furthermore the equational theory is *subterm* [6] the reduction is to a subterm deduction system as just defined above.

Now we show that if $D, \sigma, \alpha \vdash \mathcal{S}$, a term $s \in \mathrm{Sub}(D)$ is either the instance of a non-variable subterm of $\mathrm{Out}(\mathcal{S})$ or deduced by a standard composition.

Lemma 3. *Let σ be a ground substitution such that $\sigma \models \mathcal{S}$. If D is a proof of $\sigma \models \mathcal{S}$ such that no term is deduced twice in D by standard rules and s is a term such that $s \in \mathrm{Sub}(D)$ and $s \notin (\mathrm{Sub}(\mathrm{Out}(\mathcal{S})) \setminus \mathcal{X})\sigma$ **then** there exists an index i in D such that $D[i] = l \to s$ is a composition rule and $s \notin \mathrm{Sub}(\mathrm{R}_D\,(i-1))$.*

Proof. First we note that by definition of subterm deduction systems for any decomposition rule $l \to r$ we have *a)* $r \in \mathrm{Sub}(l)$, and *b)* for any composition rule $l \to r$ we have $l \subset \mathrm{Sub}(r)$ and $\mathrm{Sub}(r) \setminus \mathrm{Sub}(l) = \{r\}$.

Let D be a proof of $\sigma \models \mathcal{S}$, and let i be minimal such that $D[i] = l_r \twoheadrightarrow r$ with $s \in \mathrm{Sub}(r)$. Since $l_r \subseteq \mathrm{R}_D\,(i-1)$, the minimality of i implies $s \in \mathrm{Sub}(r) \setminus \mathrm{Sub}(l_r)$.

Thus by *a)* $D[i]$ cannot be a decomposition.

If $D[i] = ?r$ then by the $(\mathcal{S}, \sigma)$-compliance of D we have $\mathcal{S}[\alpha^{-1}(i)] = !t$ with $t\sigma = r$. We have $s \in \mathrm{Sub}(r) = \mathrm{Sub}(t\sigma) = \mathrm{Sub}(t)\sigma \cup \mathrm{Sub}(\mathrm{Var}(t)\sigma)$.

If $s \in (\mathrm{Sub}(\mathrm{Out}(S)) \setminus \mathcal{X})\sigma$ we are done, otherwise there exists $y \in \mathrm{Var}(t)$ with $s \in \mathrm{Sub}(y\sigma)$. By the origination property, there exists $k < \alpha^{-1}(i)$ such that $\mathcal{S}[k] = ?t'$ with $y \in \mathrm{Var}(t')$. Since $D, \sigma, \alpha \vdash \mathcal{S}$ and $k < \alpha^{-1}(i)$ there exists $j < i$ such that $D[j] = l_j \to t'\sigma$. The minimality of i is contradicted by $s \in \mathrm{Sub}(t'\sigma)$.

Therefore, $D[i] = l_r \to r$ is a standard composition rule. As a consequence, $\mathrm{Sub}(r) \setminus \mathrm{Sub}(l_r) = \{r\}$. Since $s \in \mathrm{Sub}(r) \setminus \mathrm{Sub}(l_r)$, we finally obtain $s = r$.

3.2 Locality

Subterm deduction systems are not necessarily local in the sense of [19]. However we prove in this subsection that given σ, there exists a finite extension T of $\mathrm{Sub}(\mathcal{S})$ and an extension σ' of σ of domain $\mathrm{Var}(T)$ and a (T, σ')-maximal derivation D in which every deduction relevant to the proof of $\sigma \models \mathcal{S}$ is liftable into a deduction between terms in T. Let us first precise the above statements.

Definition 7 (Localization set). *A set of terms T* localizes *a derivation $D = (l_i \twoheadrightarrow r_i)_{1 \le i \le m}$ for a substitution σ of domain $\mathrm{Var}(T)$ if for every $1 \le i \le m$ if $D[i]$ is a standard rule and there exists $t \in \mathrm{Sub}(T) \setminus \mathcal{X}$ such that $t\sigma = r_i$, there exists $t_1, \ldots, t_n \in \mathrm{Sub}(T)$ such that $\{t_1\sigma, \ldots, t_n\sigma\} \subseteq \mathrm{R}_D\,(i-1)$ and $t_1, \ldots, t_n \to t$ is the instance of a standard deduction rule.*

First, we prove that for subterm deduction systems, every proof D of $\sigma \models \mathcal{S}$ is localized by a set T of DAG size linear in the DAG size of $\mathcal{S}$.

Lemma 4. *If σ is a ground substitution such that $\sigma \models \mathcal{S}$ there exists $T \supseteq \mathrm{Sub}(\mathcal{S})$ of size linear in $|\mathrm{Sub}(\mathcal{S})|$, a substitution τ of domain $\mathrm{Var}(T) \setminus \mathrm{Var}(\mathcal{S})$ and a $(T, \sigma \cup \tau)$-maximal and $(\mathcal{S}, \sigma)$-compliant derivation localized by T for $\sigma \cup \tau$.*

Proof. By Lemma 2 applied with $T_1 = \emptyset$, $T_2 = \mathrm{Sub}(\mathcal{S})$, $\sigma_1 = \emptyset, \sigma_2 = \sigma$, and D_0 the $(\mathcal{S}, \sigma)$-compliant derivation that has no standard deductions, there exists a $(\mathrm{Sub}(\mathcal{S}), \sigma)$-maximal $(\mathcal{S}, \sigma)$-compliant derivation D in which no term is deduced twice by a standard deduction. From now on we let $T_0 = \mathrm{Sub}(\mathcal{S})$.

Let $\{l_i \to r_i\}_{1 \leq i \leq n}$ be the set of decompositions in D, and $\{(L_i \to R_i, \tau_i)\}_{1 \leq i \leq n}$ be a set of decomposition rules and ground substitutions such that for all $1 \leq i \leq n$ we have $L_i\tau_i \to R_i\tau_i = l_i \to r_i$. Since no term in D is deduced twice by a standard deduction, by Lemma 3 we have $n \leq |\mathrm{Sub}(\mathrm{Out}(S))|$.

Modulo variable renaming we may assume that $i \neq j$ implies $\mathrm{dom}\,(\tau_i) \cap \mathrm{dom}\,(\tau_j) = \emptyset$, and thus that $\tau = \bigcup_{i=1}^{n} \tau_i$ is defined on $T_1 = \bigcup_{i=1}^{n} (\mathrm{Sub}(L_i) \cup \mathrm{Sub}(R_i))$. Note that the size of T_1 is bounded by $M \times |\mathrm{Sub}(\mathrm{Out}(\mathcal{S}))|$, where M is the maximal size of a decomposition rule belonging to the deduction system.

Let $T = T_0 \cup T_1$ and, noting that these substitutions are defined on non-intersecting domains, let $\sigma' = \sigma \cup \tau$. By construction $|T| \leq (M + 1) \times |\mathrm{Sub}(\mathcal{S})|$.

By Lemma 2 there exists a $(\mathcal{S}, \sigma)$-compliant derivation D' which is (T, σ')-maximal and such that every deduction of a term in $T_0\sigma$ that occurs in D also occurs in D' and no term is deduced twice in D' by a standard deduction.

Let $l \to r$ be a deduction in D' which does not appear in D. Since D is (T_0, σ)-maximal we have $r \notin \mathrm{Sub}(T_0)\sigma$, and thus $r \notin \mathrm{Sub}(\mathrm{Out}(\mathcal{S}))\sigma$. Since no term is deduced twice in D' by Lemma 3 this deduction must be a composition.

Let us prove D' is (T, σ')-localized. By definition of composition rules, every composition that deduces a term $t\sigma'$ with $t \in \mathrm{Sub}(T) \setminus \mathrm{Var}(T)$ has a left-hand side $t_1\sigma', \ldots, t_k\sigma'$ with $t_1, \ldots, t_k \in \mathrm{Sub}(T)$ and $t_1, \ldots, t_k \to t$ is an instance of a composition rule. By the preceding paragraph every decomposition in D' occurs in D and thus by construction has its left-hand side in $T_1\sigma'$ which was previously built in D and is an instance of some $L_i \to R_i$ such that $\mathrm{Sub}(L_i \cup \{R_i\}) \subseteq T_1 \subseteq T$.

Thus every deduction whose right-hand side is in $(\mathrm{Sub}(T) \setminus \mathrm{Var}(T))\sigma'$ has its left-hand side in $\mathrm{Sub}(T)\sigma'$, and thus D' is localized by T for σ'.

We prove now that to solve constraint systems one can first guess equalities between terms in T and then solve constraint systems without variables. The guess of equalities is correct wrt a solution σ if terms in T that have the same instance by σ are syntactically equal. We characterize these guesses as follows.

Definition 8 (One-to-one localizations). *A set of terms T* one-to-one localizes *a derivation D for a ground substitution σ if σ is injective on* $\mathrm{Sub}(T)$ *and T localizes D for σ.*

In Lemma 7 we prove that once equalities between variables are correctly guessed there exists a one-to-one localization of a maximal proof D.

Lemma 5. *Let T be a set of terms such that $T = \mathrm{Sub}(T)$, σ be a ground substitution defined on* $\mathrm{Var}(T)$*, $U = \{p =_? q : p, q \in T \wedge p\sigma = q\sigma\}$ be a unification system and θ be its most general idempotent unifier with* $\mathrm{Var}(\mathrm{img}\,(\theta)) \subseteq \mathrm{Var}(U)$*. Then for any term t, $t\theta\sigma = t\sigma$.*

Lemma 6. *Let U be a unification system and $\theta = \mathrm{mgu}\,(U)$ an idempotent most general unifier with* $\mathrm{Var}(\mathrm{img}\,(\theta)) \subseteq \mathrm{Var}(U)$*. Then* $\forall p \in \mathrm{Sub}(\mathrm{img}\,(\theta))$ $\exists q \in \mathrm{Sub}(U) : p = q\theta$.

Lemma 7. *If σ is a ground substitution such that $\sigma \models \mathcal{S}$ **then** there exists a set of terms T, a substitution τ of domain $\mathrm{Var}(T) \setminus \mathrm{Var}(\mathcal{S})$, a substitution θ and a $(\mathcal{S}\theta, \sigma)$-compliant derivation D such that $\sigma \cup \tau = \theta(\sigma \cup \tau)$, $\mathrm{Sub}(\mathcal{S}\theta) \subseteq T$, and:*

1. *D is $(T, \sigma \cup \tau)$-maximal and one-to-one localized by T for $\sigma \cup \tau$*
2. *T and θ are of size linear in $|\mathrm{Sub}(\mathcal{S})|$*

Proof. Under the same assumptions, by Lemma 4, there exists $T_0 \supseteq \mathrm{Sub}(\mathcal{S})$ of size linear in $|\mathrm{Sub}(\mathcal{S})|$ and τ of domain $\mathrm{Var}(T_0) \setminus \mathrm{Var}(\mathcal{S})$ such that there exists a $(T_0, \sigma \cup \tau)$-maximal and $(\mathcal{S}, \sigma)$-compliant derivation D which is localized by T_0 for the same substitution $\sigma' = \sigma \cup \tau$.

Let $\mathcal{U} = \{t =_? t' : t, t' \in \mathrm{Sub}(T_0) \text{ and } t\sigma' = t'\sigma'\}$. The unification system $\mathcal{U}$ has a unifier σ' and thus has a most general solution θ. By Lemma 5, $\sigma' = \theta\sigma'$.

Let $T = \mathrm{Sub}(T_0)\theta$.

Since $\mathrm{Sub}(\mathcal{S}) \subseteq T_0$ we have $\mathrm{Sub}(\mathcal{S}\theta) \subseteq \mathrm{Sub}(T_0\theta)$. Since θ is a most general unifier of $\mathcal{U}$ and $\mathrm{Sub}(\mathcal{U}) = \mathrm{Sub}(T_0)$ we have $\mathrm{Sub}(T_0\theta) = \mathrm{Sub}(T_0)\theta$ by Lemma 6. This implies (i) $\mathrm{Sub}(\mathcal{S}\theta) \subseteq T$, (ii) θ is of linear size on $|\mathrm{Sub}(T_0)|$ and thus on $|\mathrm{Sub}(\mathcal{S})|$, and (iii) T is of linear size on $|\mathrm{Sub}(\mathcal{S})|$. Moreover, as $\sigma' = \theta\sigma'$ we have $\mathrm{Sub}(T)\sigma' = \mathrm{Sub}(T_0)\sigma'$ and thus from D is (T_0, σ')-maximal follows D is (T, σ')-maximal.

Assume there exists $t, t' \in \mathrm{Sub}(T)$ such that $t\sigma' = t'\sigma'$ but $t \neq t'$. Since $T = \mathrm{Sub}(T_0\theta)$ there exists $t_0, t'_0 \in \mathrm{Sub}(T_0)$ such that $t_0\theta \neq t'_0\theta$ but $t_0\theta\sigma' = t'_0\theta\sigma'$. From $\sigma' = \theta\sigma'$ we have an existence of $t_0, t'_0 \in \mathrm{Sub}(T_0)$ such that $t_0\theta \neq t'_0\theta$ but $t_0\sigma' = t'_0\sigma'$. This contradicts the fact that θ satisfies $\mathcal{U}$.

Finally, from D is $(\mathcal{S}, \sigma)$-compliant and $\sigma = \theta\sigma$ we have D is $(\mathcal{S}\theta, \sigma)$-compliant.

3.3 Milestone Sequence

In addition to retrace the deduction steps performed in D we want to track which terms relevant to $\mathcal{S}$ are deduced in T, and in which order.

Definition 9 (Milestone sequence). *Let T be a set of terms and σ be a ground substitution. We say that $\boldsymbol{T}$ is the (T, σ)-milestone sequence of a derivation $D = (l_i \to r_i)_{1 \leq i \leq m}$ if $\boldsymbol{T} = t_1, \ldots, t_n$ is a sequence of maximal length in which each t_i is either of the form $\to t$ or of the form $?t$, with $t \in \mathrm{Sub}(T)$ and there exists a strictly increasing function $\alpha : \{1, \ldots, n\} \to \{1, \ldots, m\}$ such that for every $1 \leq i \leq n$ we have:*

1. *if $\boldsymbol{T}[i] = ?t$ then $D[\alpha(i)] = ?t\sigma$;*
2. *if $\boldsymbol{T}[i] = \to t$ then $D[\alpha(i)] = l_i \to t\sigma$ is a standard deduction rule;*

Lemma 8. *Let $\sigma \models \mathcal{S}$, $T \supseteq \mathrm{Sub}(\mathcal{S})$ and σ' be an extension of σ on $\mathrm{Var}(T)$. Let D be (T, σ')-maximal derivation one-to-one localized by T for σ'. Let $\boldsymbol{T}$ be a (T, σ')-milestone sequence. **Then** for any i for any $x \in \mathrm{Var}(\boldsymbol{T}[i])$ there exists $j < i$ such that $\boldsymbol{T}[j] = \to x$.*

Proof. If $x \in \mathrm{Var}(\boldsymbol{T}[i])$ then there exists corresponding deduction $D[j]$ that deduces term $\boldsymbol{T}[i]\sigma'$. Then by Lemma 3 there exists $k < j$ such that $D[j]$ deduces

by a standard rule $x\sigma'$. From the injectivity of σ follows that x is the only term of $\mathrm{Sub}(T)$ having σ' image equal $x\sigma'$. Thus, by definition of milestone sequence, there exists $m < i$ such that $\boldsymbol{T}[m] =\!\!\rightarrow x$.

4 Deciding Constraint Systems

From now we suppose that the **considered subterm deduction system** contains a rule $x_1, x_2 \to f(x_1, x_2)$, where f is a function symbol with arity 2 that does not occur in any other rule.

Theorem 1. *Let σ such that $\sigma \models \mathcal{S}$, T such that $T \supseteq \mathrm{Sub}(\mathcal{S})$ and σ' an extension of σ on $\mathrm{Var}(T)$. Let D be a (T, σ')-maximal derivation one-to-one localized by T for σ' in which no term is deduced twice by a standard rule.*

Then there exists a solution τ of $\mathcal{S}$ of size polynomial in $|\mathrm{Sub}(T)|$.

Proof (sketch, for formal proof see [4]). Let $\boldsymbol{T}$ be a (T, σ')-milestone sequence for D.

We prove the existence of a ground substitution τ', set of terms $T' \supseteq T$ and a derivation D' which is $(\mathcal{S}, \tau)$-compliant, (T', τ)-maximal (where $\tau = \tau'|_{\mathrm{Var}(\mathcal{S})}$ is of a linear size on $\mathrm{Sub}(T)$) and is one-to-one localized by T' with τ' such that its (T, τ')-milestone sequence coincides with the $\boldsymbol{T}$. Having this proved, by Lemma 1 we can show that $\tau \models \mathcal{S}$.

Let X be such subset of $\mathrm{Var}(\boldsymbol{T})$ whose σ'-instance are not derivable from the empty knowledge. We define T' by adding to T one fresh variable $\bar{x}$ per each variable x of X.

We define a ground substitution τ' for any $y \in \bar{X} \cup (\mathrm{Var}(\boldsymbol{T}) \setminus X)$ as a fresh nonce; and for any $x \in X$, we put, thanks to Lemma 8, $x\tau' = f(t_x, \bar{x})\tau'$, where t_x is the last term received ($?t_x$) before the first occurence of $\to x$ in $\boldsymbol{T}$.

We can see that $x\tau$ is of polynomial size on $|\mathrm{Sub}(\mathcal{S})|$ for any $x \in \mathrm{Var}(T)$.

By such construction, and particulary since τ' is injective on $\mathrm{Var}(\boldsymbol{T})$, we can show that τ' is also injective on $\mathrm{Sub}(T')$.

Then we construct a $(\mathcal{S}, \tau)$-compliant derivation D' localized by T' with τ'. First $|\bar{X}|$ rules of D' are generations of nonces $(\bar{X})\tau'$. Then, for each element $!t$ (resp. $\to t$) of $\boldsymbol{T}$ we add $!t\tau'$ (resp. a rule that deduces $t\tau'$; we can do it since $\boldsymbol{T}$ origins from D which is one-to-one localized by T) in D'. Moreover, D' is one-to-one localized since τ' is injective on $\mathrm{Sub}(T')$.

We can show that D' is (T', τ')-maximal, i.e. for any $t \in \mathrm{Sub}(T')$ if $t\tau' \in \mathrm{Der}(\mathrm{R}_{D'}(i))$ then $t\tau' \in \mathrm{R}_{D'}(\mathrm{Next}_{D'}(i) - 1)$.

Suppose by contradition, there exists t not satisfying the maximality condition for minimal possible i. If t is a variable not in $\bar{X}$, then, by construction of τ' we could also derive $t\sigma'$ from the corresponding point of D and thus $\to t$ should have been in $\boldsymbol{T}$ and $t\tau'$ must be deduced in a "good" (w.r.t. maximality) position in D'. Note that case of $t \in \bar{X}$ is trivial.

If t is not a variable, we can consider derivation E proving that $t\tau'$ is derivable from $D'[1 : i]$. Then, doing a replacement of term $x\tau'$ by $x\sigma'$ (for each variable x) in E, we obtain a proof that $t\sigma'$ can be derived from $D[1 : (i - |\bar{X}|)]$.

Summing up, since $\mathrm{Sub}(\mathcal{S}) \subseteq T'$, *and* τ' *is injective on* $\mathrm{Sub}(T')$, *we have that by construction of* D', *for any term* $t \in \mathrm{Sub}(\mathcal{S})$, $t\sigma'$ *is deduced before* j*-th non-standard rule of* D *(resp. deduced in* D*) if and only if* $t\tau'$ *is deduced before* j*-th non-standard rule of* D' *(resp. deduced in* D'*). Therefore, since* $\sigma \models \mathcal{S}$ *and* D *is* $(\mathcal{S}, \sigma)$*-compliant and* $(\mathrm{Sub}(\mathcal{S}), \sigma)$*-maximal and since* D' *is* $(\mathcal{S}, \tau)$*-compliant and* $(\mathrm{Sub}(\mathcal{S}), \tau)$*-maximal we may use twice Lemma 1 and obtain that* τ *satisfies* $\mathcal{S}$.

Corollary 1. *Let* $\mathcal{S}$ *be a constraint system.* $\mathcal{S}$ *is satisfiable, if and only if there exists a solution* σ' *of* $\mathcal{S}$ *with polynomial size w.r.t.* $|\mathrm{Sub}(\mathcal{S})|$.

Proof. $(\Leftarrow)$ *is trivial, since* $\sigma' \models \mathcal{S}$. *Consider* $(\Rightarrow)$. *Let* $\sigma \models \mathcal{S}$. *By Lemma 7 there exists a set of terms* T, *a substitution* θ *both with the size linear in* $|\mathrm{Sub}(\mathcal{S})|$ *and an extension* γ *of* σ *and* (T, γ)*-maximal* $(\mathcal{S}\theta, \sigma)$*-compliant derivation* D *one-to-one localized by* T *for* γ. *We also have* $\gamma = \theta\gamma$ *(which implies* $\sigma = \theta\sigma$*). Thus* σ *satisfies* $\mathcal{S}\theta$.

From the same lemma we have $\mathrm{Sub}(\mathcal{S}\theta) \subseteq T$. *By Theorem 1 there exists a substitution* τ *of size polynomial in* $|\mathrm{Sub}(T)|$ *(and consequently, polynomial in* $|\mathrm{Sub}(\mathcal{S})|$*) such that* $\tau \models \mathcal{S}\theta$. *From this we have* $\theta\tau \models \mathcal{S}$. *Moreover, since both* θ *and* τ *are of polynomial size on* $|\mathrm{Sub}(\mathcal{S})|$, $\sigma' = \theta\tau$ *is also of polynomial size on* $|\mathrm{Sub}(\mathcal{S})|$ *and* $\sigma' \models \mathcal{S}$.

From the previous result we can directly derive an NP decision procedure for constraint systems satisfiability: guess a substitution of polynomial size in $|\mathrm{Sub}(\mathcal{S})|$ and check whether it satisfies $\mathcal{S}$ in polynomial time (see e.g. [1]). We also recall that the problem of solving a system of deducibility constraints is NP-complete [21,23] and since the considered in this paper problem is more general, we conclude its NP-completeness.

5 Conclusion

We have obtained the first NP-decision procedure for deducibility constraints with negation and we have applied it to the synthesis of mediators subject to non-disclosure policies. It has been implemented as an extension of CL-AtSe [24] for the Dolev-Yao deduction system. On the Loan Origination case study, the prototype generates directly the expected orchestration. Without negative constraints undesired solutions in which the mediator impersonates the clerks were found. More details about the implementation and tested specifications can be found at `http://cassis.loria.fr/Cl-Atse`.

As in [1,6] our definition of subterm deduction systems can be extended to allow ground terms in right-hand sides of decomposition rules even when they are not subterms of left-hand sides and the decidability result remains valid with minor adaptation of the proof. A more challenging extension would be to consider general constraints (as in [3]) with negation.

The full version of proofs presented in the paper can be found in [4].

References

1. Abadi, M., Cortier, V.: Deciding knowledge in security protocols under equational theories. Theoretical Computer Science 367(1-2), 2–32 (2006)
2. Avanesov, T., Chevalier, Y., Mekki, M.A., Rusinowitch, M.: Web Services Verification and Prudent Implementation. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cuppens-Boulahia, N., de Capitani di Vimercati, S. (eds.) DPM 2011 and SETOP 2011. LNCS, vol. 7122, pp. 173–189. Springer, Heidelberg (2012)
3. Avanesov, T., Chevalier, Y., Rusinowitch, M., Turuani, M.: Satisfiability of general intruder constraints with and without a set constructor. CoRR, abs/1103.0220 (2011)
4. Avanesov, T., Chevalier, Y., Rusinowitch, M., Turuani, M.: Intruder deducibility constraints with negation. Decidability and application to secured service compositions. INRIA Research Report (July 2012), http://hal.inria.fr/hal-00719011
5. Automated Validation of Trust and Security of Service-Oriented Architectures, AVANTSSAR project, http://www.avantssar.eu
6. Baudet, M.: Deciding security of protocols against off-line guessing attacks. In: Proceedings of CCS 2005 Conference, pp. 16–25. ACM (2005)
7. Chevalier, Y., Mekki, M.A., Rusinowitch, M.: Automatic composition of services with security policies. In: Proceedings of SERVICES I 2008, SERVICES 2008. pp. 529–537. IEEE, Washington, DC (2008)
8. Corin, R., Etalle, S., Saptawijaya, A.: A logic for constraint-based security protocol analysis. In: IEEE Symposium on Security and Privacy (S&P), Berkeley, California, USA, May 21-24, pp. 155–168. IEEE Computer Society (2006)
9. Costa, G., Degano, P., Martinelli, F.: Secure service orchestration in open networks. Journal of Systems Architecture - Embedded Systems Design 57(3), 231–239 (2011)
10. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
11. Armando, A., Arsac, W., Avanesov, T., Barletta, M., Calvi, A., Cappai, A., Carbone, R., Chevalier, Y., Compagna, L., Cuéllar, J., Erzse, G., Frau, S., Minea, M., Mödersheim, S., von Oheimb, D., Pellegrino, G., Ponta, S.E., Rocchetto, M., Rusinowitch, M., Torabi Dashti, M., Turuani, M., Viganò, L.: The AVANTSSAR Platform for the Automated Validation of Trust and Security of Service-Oriented Architectures. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 267–282. Springer, Heidelberg (2012)
12. Frau, S., Torabi Dashti, M.: Integrated specification and verification of security protocols and policies. In: 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, June 27-29, pp. 18–32 (2011)
13. Herzig, A., Lorini, E., Hübner, J.F., Vercouter, L.: A logic of trust and reputation. Logic Journal of IGPL 18(1), 214–244 (2010)
14. Kourjieh, M.: Logical Analysis and Verification of Cryptographic Protocols. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, (Décembre 2009)
15. Kourjhler, D., Ksters, R., Truderung, T.: Infinite state amc-model checking for cryptographic protocols. In: Symposium on Logic in Computer Science, pp. 181–192 (2007)
16. Lorini, E., Demolombe, R.: Trust and Norms in the Context of Computer Security: A Logical Formalization. In: van der Meyden, R., van der Torre, L. (eds.) DEON 2008. LNCS (LNAI), vol. 5076, pp. 50–64. Springer, Heidelberg (2008)

17. Lynch, C., Meadows, C.: On the relative soundness of the free algebra model for public key encryption. In: Proceedings of the 2007 FCS-ARSPA Workshop. ENTCS, vol. 125, pp. 43–54 (2005), http://profs.sci.univr.it/~vigano/fcs-arspa07/fcs-arspa07.pdf
18. Martinelli, F.: Towards an Integrated Formal Analysis for Security and Trust. In: Steffen, M., Zavattaro, G. (eds.) FMOODS 2005. LNCS, vol. 3535, pp. 115–130. Springer, Heidelberg (2005)
19. McAllester, D.A.: Automatic recognition of tractability in inference relations. Journal of the ACM 40, 284–303 (1993)
20. Millen, J., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001, pp. 166–175. ACM, New York (2001)
21. Millen, J.K., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: Proceedings of the ACM Conference on Computer and Communications Security CCS 2001, pp. 166–175 (2001)
22. Network of Excellence on Engineering Secure Future Internet Software Services and Systems, NESSoS project, http://www.nessos-project.eu
23. Rusinowitch, M., Turuani, M.: Protocol insecurity with finite number of sessions is NP-complete. In: Proceedings of CSFW 2001, pp. 174–190. IEEE Computer Society Press (2001)
24. Turuani, M.: The CL-Atse Protocol Analyser. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 277–286. Springer, Heidelberg (2006)
25. Chevalier, Y., Mekki, M.A., Rusinowitch, M.: Orchestration under Security Constraints. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) FMCO 2010. LNCS, vol. 6957, pp. 23–44. Springer, Heidelberg (2011)

An Approach for Network Information Flow Analysis for Systems of Embedded Components

Andrey Chechulin, Igor Kotenko, and Vasily Desnitsky

St. Petersburg Institute for Informatics and Automation (SPIIRAS)
39, 14 Linija, St. Petersburg, Russia
{chechulin,ivkote,desnitsky}@comsec.spb.ru

Abstract. Systems (devices) with embedded components operate in a potentially hostile environment and have strong recourse limitations. The development of security-enhanced embedded components is a complicated task owning to different types of threats and attacks that may affect the device, and because the security in embedded devices is commonly provided as an additional feature at the final stages of the development process, or even neglected. In the paper we consider an approach to analysis of network information flows in systems containing embedded components. This approach helps to the system engineer to evaluate the embedded system from security point of view and to correct the architecture of future system on early stages of the development.

Keywords: information flows, embedded devices, model checking, topology analysis.

1 Introduction

The paper encompasses the analysis of security issues of the systems which include embedded devices. Such systems are notable for autonomy of separate devices included in the system and for constrains of the resources of the device and their consequently week efficiency [1-4].

The notion of the information flow is very important for information security as it describes how information spreads and who accesses it in the systems with embedded components. The standard method to protect confidential data is to control access to it: only entities with specific privileges can access files or objects containing confidential data. But this approach does not consider the covert channels emerging when the information is propagated in the information system either through the program execution or through the hardware processing. The usage of the information flows help to track the information spread. Information flow security is based on the concept of the non-interference that formalizes the relationships between components of the system [5]. A noninterference security policy specifies which components, or domains, may not interfere with each other; it determines where component u interferes with a component v if v can observe the effects of u's execution. This means that no secret data could be obtained by observing "public" behavior of the system. The non-interference model mainly focuses on the events and system actions and their

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 146–155, 2012.

impact on the secrecy of the confidential data. D. Oheimb [6] proposes the notions of *nonleakage* and *noninfluence*. *Nonleakage* focuses primarily on the secret state of the system while *noninfluence* is the combination of the *nonleakage* and noninterference.

B. Lampson [7] defines three types of channels which can cause unsafe information flows. The first one is the *legitimate channels* that are intended for information transfer when legitimate system operation is implemented. Another type of the channels is the *storage channel* designed to store information. The last type of the channels consists of so-called *covert channels* that emerge as the result of the side-effects of legitimate mechanisms. The aim of information flow analysis is to reveal the existence of such channels in the information system.

The interference concept is the basis for the security analysis of information flows, producing security rules and requirements to the information exchange. In general, information flow analysis of a security-critical system can be done in either of two ways: (1) *Dynamically*, by tracking how information actually flows through the system when it is in operation. This is often easy to implement, by inserting sensors and monitors into the system, but has the disadvantage that even if no insecure flows have been observed to date we can never be certain that no security problems will arise in future; (2) *Statically*, by analyzing the structure of the system itself. This has the advantage of providing absolute guarantees about all of the system's potential behaviors, but it is often computationally expensive and produces "false-positives", i.e. alerts about potential information flows identified statically that never actually occur dynamically.

The essential idea of static information flow analysis is to allow an information security evaluator to see where classified data could propagate when the system is in operation. This is usually done by selecting some point in the system to be a source of high-security data, and then using a connectivity graph model of the system to trace possible pathways for this data. This is sometimes referred to as "taint analysis" and is helpful in evaluating a system's weak points and mechanisms for protecting data confidentiality and integrity. The primary disadvantages of the approach are that it is computationally expensive for large system models and tends to overapproximate the possible data flow pathways, resulting in false-positive results that can waste an security evaluator's time (although this is preferable to underestimating potential data flow, which could result in security-critical pathways being overlooked).

Security-critical information flow analysis can be applied at various levels, including: (1) schematic diagrams of electronic circuitry (hardware flows); (2) data flow models of computer software (software flows); and (3) entire communication network layouts (network flows). Data flow analysis of computer software has been explored extensively [8-10], but similar analyses for embedded system hardware and network designs have received relatively little attention.

The approach proposed in the paper determines a way of simultaneous use of combination of particular algorithms and techniques implementing various methods for information flow analysis. The goal of this approach is to achieve high security level by analyzing the architecture and implementation of the system with embedded components. The main difference of the offered approach from the already suggested ones is the integration of these functionalities in one component to achieve better results. The approach novelty consists also in the way of applying the existing methods for different layers of information flow model. The paper is focused mainly on network information flows analysis. It is performed in SecFutur project [11].

The rest of this paper is organized as follows. Section 2 is devoted to related work analysis. Section 3 presents suggested methodology for information flow analysis on the network layer. In section 4 we show how suggested methodology could be applied to a real use-case. Finally, section 6 presents some conclusions and future work.

2 Related Work

While many techniques exist to track information flows through software, little work has been done in the analysis of the network information flow security. The notion of the information flow is widely used when estimating routing capacity, assessing network efficiency or constructing networks [12-14]. Though this research is not directly related to the information flow non-interference in the network it could be useful when modelling and constructing information flows. The information flow between nodes is usually represented as directed acyclic graphs, where each node corresponds to a point-to point communication host, and edges reflect information flows. The topological analysis can be applied to reveal covert channels [15]. A number of papers is devoted to the detection mechanisms of covert channels which are based on the network protocol features [16-17].

V. Shnayder [18] examines the opportunities for applying language based security techniques, in particular information flow tracking, to the sensor network domain. D.P.Grushka [19] presents a formal model for the analysis of information flows in networks. It is based on the timed process algebra [20], which can express also network properties. The presented calculus allows modelling of two types of communications (via fast networks, for example local buses, and via shared networks with limited throughput, for example optical networks with wave length-division multiplexing) and modelling of complex networks combining both types of communications. This approach enables to formalize the notion of "network timing attacks", to reveal timing covert channels attack and to modify the system in order to eliminate timing attacks.

Network security policies can be used for description of the information flows allowed or prohibited in the system. Thus, their analysis can be helpful in the analysis of information flows in the network. There are different approaches for analysis and verification of the security policies, the most of them are based on theorem proving and model checking, though special-purpose techniques developed for verification of the specific policy rules also exist. There are many works in detecting misconfiguration in network access control devices such as firewalls, routers [21-23].

Other works present general models for analysing network configuration. R.Bush and T.Griffin [24] suggest a formal model of the VPN, which uses BGP to propagate routing information for all VPNs implemented within a provider's backbone and a tunneling technology, such as MPLS, to isolate traffic. In particular authors focus on integrity constraints that must be maintained by providers in order to ensure that intra-VPN connectivity is achieved, and that disjoint VPNs are isolated.

Al Shaer et al. [25] present a model of the network based on the state machine where the packet header and location determine the state. The transitions in this model are determined by the packet header information, the packet location, and the

policy semantics for the devices. The semantics of access control policies is encoded with Boolean functions using binary decision diagrams [26].

R.Bryant [27] uses tree logic and symbolic model checking to investigate all future and past states of the packet in the network computation and to verify network reachability and security requirements. The author demonstrates the effectiveness of the model through the provability of soundness and completeness of the configuration reachability as well as discovering number of security violation examples such as backdoors and broken IPSec tunnels. The proposed model is implemented in a tool called ConfigChecker [28].

3 Approaches for Information Flow Analysis

In this paper two main approaches to network information flow analysis are supported – topological and policy based. *Topological* analysis takes as input the directed graph and evaluates different path from security point of view. *Policy based* approach works with policies, which describe information flows in the formal format, and tries to find contradictions between them. Let us consider these approaches in more detail.

3.1 Topological Information Flow Analysis Principles

Simple topological analysis techniques, based on the conventional graph theory, can be of significant help to an information security evaluator. In particular, given a model of a security-critical system as a directed graph of nodes (vertices) and arcs (edges), the following kinds of analyses can be helpful when trying to evaluate the system's security characteristics with respect to the potential information flows.

1. *Identifying all components that lie between two selected points in the graph, typically between a high-security information source and a low-security sink.* An information security evaluator can use this kind of analysis to identify the "security perimeter" or "security-critical region" within the system, in order to quickly eliminate those components that have no security significance [15]. Components that do not appear between the source and sink cannot play a role in the security argument and do not need to be analyzed further.

2. *Most importantly, finding all data flow paths between two selected points in a graph, typically between a high-security information source and a low-security data sink.* This provides an information security evaluator with a set of pathways worthy of close inspection, to determine whether or not they are allowable in the context of the intended security behavior of the system. Unfortunately, finding all paths between two points in a graph cannot be done efficiently for large graphs because it subsumes the problem of finding the longest path, which is known to be NP-complete. Therefore, a divide-and-conquer strategy must be employed to analyze large systems.

Of these techniques, the one that is most commonly helpful to an information security evaluator is the last, i.e., finding all the possible information flow paths between a selected high-security data source and a low-security data sink. Once the paths have been found, the information security evaluator can then inspect each of them to

determine whether or not they pose a threat to system security. For each such path there is a number of possible outcomes from this assessment. In increasing order of usefulness these are as follows: (1) The path was one expected by the information security evaluator, helping confirm the anticipated normal behavior of the system; (2) The path was one expected by the information security evaluator, but represents an undesirable property of the system, thus confirming that the system has a known problem; (3) The path was one not expected by the information security evaluator but, upon close inspection, proves to be a valid one for the system, thus helping the evaluator understand the system's 'normal' behavior; (4) The path was one not expected by the information security evaluator and, upon close analysis, proves to be an undesirable one, thus revealing a previously-unknown security problem.

However, for non-trivial system models the number of paths generated can be very high, with many of them being false-positives, making this process highly tedious and thus error-prone. One way of reducing the number of false positives is to introduce some simple static semantics into the model. One approach for doing this is to associate known system operating modes with the possible data flows through components [15]. This allows a mode-specific path analysis to be performed in which impossible paths, where data travels through components in contradictory modes, can be immediately eliminated. Another possibility is to group pathways together so that several can be assessed at once.

3.2 Information Flows Analysis Based on Security Policies

There are two approaches to perform network analysis for information flows: static and dynamic approach.

The main objective of the *static approach* is to check conformity of the formally described physical and logical connections between network elements to the policy flow model. It can be checking the presence of strong encryption, checking the presence of mutual authentication, checking network flow policies for collisions, etc.

The *dynamic approach* in turn is targeted to check real data flows detected in the network flows for conformity to the policy flow model. Information about real connections in the network can be collected from the following elements: connections at the network level (TCP sessions); connections at the application level (VPN tunnels, etc.); by assigning labels to the data and tracing them.

For information flow analysis on the network level the Flow Security Language (FSL) is proposed. Flows are specified in this language using the following parameters: users, end nodes, nodes through which the flow passes, the direction of the flow, and the type of data that it transfers.

For verification of the policy rules describing information flows the Model checking approach is used [30-31].

The essence of model checking is to iterate states in which the system can transfer. The state is changed according to the emerging information flow. The sequence of steps in the iteration depends on the conditions that are formulated in the linear temporal logic language and define the correct state of the system. A state of the system is determined by a set of variables, while a change of the state is caused by internal pro-

cesses. The process, which should be carried out in the next time point is chosen randomly. The system considers all possible sequences of steps and signals on the potential incorrect states. After that, the user receives "route", i.e. a sequence of steps leading to the incorrect system's state.

The main inputs for the proposed approach are: (1) description of filtering rules (policies) for information flows; (2) configuration of the network containing embedded devices; (3) description of rule`s anomalies.

At the first stage, the input data containing the description of policies and anomalies are transformed to the internal format of the verification system. Then, at the second stage, the common model for rules verification is formed by means of the finite state machine. This model is initialized with the input data in the internal format. Anomalies in the model are represented as formal statements. For the model checking approach these statements are correction properties, which violation will transfer the system into an incorrect state. At the third stage, the common model for rules verification is checked by special software tools implementing the model checking approach. In the process of verification all incorrect states of the system are identified. At the final stage of the verification all results are interpreted. If anomalies are detected, then a description of the situation (containing an information flow specification) is created.

Special software tools that implement the model checking method can be used to verify the rules of security policy of information flows, for instance, SPIN [32], SMV [33], MOCHA [34], etc. In the verification process these tools allow identifying all the conflicts between rules for the information flow control. We use the software tool SPIN and the specification language PROMELA for policies verification. PROMELA (a PROcess MEta LAnguage) is a high level language used in SPIN which allows specifying systems descriptions. The main entities such as information flow control rules, information flows, anomalies, etc. are described as data structures in this language. This structure allows storing all rule's parameters: description of the action which defines the rule for information flows (allow, deny), the name of the rule, the type of the transmitting data, the parameters of the source point (user, node, interface), the parameters of the destination point (user, node, interface), and the rule status (enabled, disabled).

The policy structure contains the policy name, the policy type and the set of rules. Data exchange between processes in the PROMELA language occurs through message channels. For linear temporal logic formulas in PROMELA language a special keyword *assert* (formal approval) is used. In the process of verification each violation of these formal statements is identified and an example showing how the system goes to an incorrect state is constructed. Thus, all the rules that can cause the anomalies are identified. Information flow policies and rules are created during initialization. Further main processes, such as information flow generation process (generateInformationFlow), information flow control process (informationFlowManager) and anomaly detecting process (detectFilteringConflicts), are started. After the entire range of hosts, users, interfaces, and types of flows are processed, the verification is completed. The results of verification activity are all violations of formal statements, i.e. detected rule's anomalies.

4 Case Study and Experiments

As use case example the Smart Metering Devices Network was chosen. The Smart Metering Devices Network is an advanced metering infrastructure consisting of several trusted meters, database servers, client applications and communication infrastructure. Its purpose is to measure energy consumption of households and to facilitate the assignment of consumption data to customers for billing purposes. It supports different user roles in a way which reflects the typical organization of the parties involved in the business, thereby providing intuitive functions for its users, who are able to interact with the components of the system via local and remote interfaces.

We implemented a software prototype which allows us to experiment with verification of network information flows for this case study.

The examples of specification of information flow control rules for Metering Device case study are shown in Listing 1 and Listing 2.

Listing 1. Main data structures.

```
typedef FilteringRule {
  bool action = true; /*true – allow, false – deny*/
  mtype ruleName;
  mtype = {any, manufacturer, calibrator, technician,
    TSN_administrator, operator_TRM, Operator _OAB,
    operator_administrator, customer };
  chan sourceUser = [1] of {mtype};
  chan destinationUser = [1] of {mtype};
  mtype = {any, operatorPC, TSNPC, RemotePC, OCSS, TSNS,
    Gateway, GPT, TSMC, TSM, TS };
  chan sourceNode = [1] of {mtype};
  chan destinationNode = [1] of {mtype};
  mtype = {any, TSMLI, TSMCLI, TSMNI, TSMCNI, SSI, GI, PCI, DBI };
  chan sourceInterface = [1] of {mtype};
  chan destinationInterface = [1] of {mtype};
  mtype = {any, Customer_account_data, Privacy_non_relevant_data,
    Privacy_relevant_consumption_data,
    Manufacturer_certificate, Calibration_certificate,
    Administrator_user_account_data, Operator_user_account_data,
    Installation_certificate, Deinstallation_certificate,
    Communication_configuration, Functional_settings,
    Security_settings, Event_records, Trusted_records, Time,
    Installed_software_and_related_resources};
  chan dataType = [1] of {mtype};
  mtype = {Enabled, Disabled};
  chan en = [1] of {mtype};
}
```

Listing 2. Specification of information flows.

```
/***********MODEL INITIALIZATION*************/
mtype = {filtering_rule_1, filtering_rule_2, filtering_rule_3, …};
proctype initModel () {
/***Creation rules***/
 FilteringRule fRule1;
 FilteringRule fRule2;
/***Init rules***/
 fRule1.ruleName = filtering_rule_1;
 fRule1.sourceUser! TSN_administrator;
 fRule1.sourceNode! TSNS;
 fRule1.sourceInterface! any;
 fRule1.destinationUser! any;
 fRule1.destinationNode! TSMC;
 fRule1.destinationInterface! any;
 fRule1.dataType! Communication_configuration;
 fRule1.action = true;
 fRule1.en!Enabled;
 fRule2.ruleName = filtering_rule_2;
 fRule2.sourceUser! any;
 fRule2.sourceNode! TSMC;
 fRule2.sourceInterface! any;
 fRule2.destinationUser! any;
 fRule2.destinationNode! OCSS;
 fRule2.destinationInterface! any;
 fRule2.dataType! Privacy_relevant_consumption_data;
 fRule2.action = true;
 fRule2.en!Enabled;
```

Listing 1 specifies the main data structures: action, user, interface, node and flow types. Listing 1 shows a description of two information flows. The first defines flow which initiated by TSN (Trusted Sensor Network) administrator from TSNS (Trusted Sensor Network Admin Server) node and which goes to TSMC (Trusted Sensor Module Collector) node and consists management information. Second flow consists of measures and goes from TSMC (Trusted Sensor Module Collector) to OCSS (Operator Collector Storage Server).

The experiments conducted showed that the proposed approach can detect all anomalies in information flow control rules, but can be used effectively only having some limited number of rules. Adding temporal characteristics to the rules have not changed significantly the time of verification.

5 Conclusion

Several approaches were analyzed and implemented to perform the analysis of information flows on network layer. For all of them the state-of-the-art was studied, and the approaches for information flow analysis, which are applicable to testing processes,

were analyzed. The smart metering device use case was elaborated. This paper is mainly devoted to theoretical aspects of information flow analysis. A deeper practical application of these approaches requires additional series of experiments.

We compared our approach with existing ones and found that it allows us to achieve more complex evaluation of information flow security in contrast to separated approaches. In future work it is planned to perform series of experiments for other use cases. Also we are going to elaborate the software prototype which will contain all described approaches to information flow analysis including hardware and software flows analysis.

Acknowledgements. This research is being supported by grant of the Russian Foundation of Basic Research (project #10-01-00826-a), Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (contract #2.2), State contract #11.519.11.4008 and partly funded by the EU as part of the SecFutur and MASSIF projects.

References

1. Desnitsky, V., Kotenko, I., Chechulin, A.: An Abstract Model for Embedded Systems and Intruders. In: Proceedings of the Work in Progress Session Held in Connection with the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2011), pp. 25–26. SEA-Publications, SEA-SR-29 (2011)
2. Desnitsky, V., Chechulin, A.: Model of the Process for Secure Embedded Systems Development. High Availability Systems (2), 97–101 (2011) (in Russian)
3. Kotenko, I., Desnitsky, V., Chechulin, A.: Investigation of Technologies for Secure Embedded Systems Design in European Union Project SecFutur. Information Security Inside (3), 68–75 (2011) (in Russian)
4. Desnitsky, V., Kotenko, I., Chechulin, A.: Constructing and Testing Secure Embedded Systems. In: Selected Proceedings of XII Saint-Petersburg International Conference "Regional informatics-2010" ("RI-2010"), pp. 115–121. St. Petersburg (2011) (in Russian)
5. Rushby, J.: Noninterference, Transitivity, and Channel-control Security Policies, SRI International. Tech. Rep. CSL-92-02 (1992)
6. von Oheimb, D.: Information Flow Control Revisited: Noninfluence = Noninterference + Nonleakage. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 225–243. Springer, Heidelberg (2004)
7. Lampson, B.: A note on the confinement problem. Communications of ACM 16(10), 613–615 (1973)
8. Pistoia, M., Chandra, S., Fink, S., Yahav, E.: A Survey of Static Analysis Methods For Identifying Security Vulnerabilities in Software Systems. IBM Systems Journal 46(2), 265–288 (2007)
9. Hedin, D., Sabelfeld, A.: A Perspective on Information-Flow. Summer school Control Tools for Analysis and Verification of Software Safety and Security, Marktoberdorf, Germany (2011)
10. Sabelfeld, A., Myers, A.C.: Language-based Information-flow Security. IEEE Journal on Selected Areas in Communications 21(1), 5–19 (2003)
11. SecFutur project website, `http://secfutur.eu`
12. Ahlswede, R., Cai, N., Li, S.-Y.R., Yeung, R.W.: Network Information Flow. IEEE Transactions on Information Theory IT-46(4), 1204–1216 (2000)

13. Sprintson, A., El Rouayheb, S., Georghiades, C.: A New Construction Method for Networks from Matroids. In: Proceedings of the 2009 IEEE International Conference on Symposium on Information Theory (ISIT 2009), Seoul (2009)
14. Agaskar, A., He, T., Tong, L.: Distributed Detection of Multi-hop Information Flows with Fusion Capacity Constraints. IEEE Transactions on Signal Processing 58(6), 3373–3383 (2010)
15. Rae, A., Fidge, C.: Information Flow Analysis for Fail-Secure Devices. The Computer Journal 48(1), 17–26 (2005)
16. Cabuk, S., Brodley, C.E., Shields, C.: IP Covert Channel Detection. ACM Transactions on Information and System Security (2008)
17. Berk, V., Giani, A., Cybenko, G.: Detection of Covert Channel Encoding in Network Packet Delays. Technical Report TR536 (2005)
18. Shnayder, V.: Opportunities for Language Based Information Flow Security in Sensor Networks (2004)
19. Gruska, D.P.: Network Information Flow. Fundamentae Informaticae 72(1-3), 167–180 (2006)
20. Gruska, D.P., Maggiolo-Schettini, A.: Process Algebra for Network Communication. Fundamenta Informaticae 45(4), 359–378 (2001)
21. Al-Shaer, E., Hamed, H., Boutaba, R., Hasan, M.: Conflict Classification and Analysis of Distributed Firewall Policies. IEEE Journal on Selected Areas in Communications (JSAC) 23(10) (2005)
22. Al-Shaer, E., El-Atawy, A., Samak, T.: Automated Pseudo-live Testing of Firewall Configuration Enforcement. IEEE Journal on Selected Areas in Communications 27(3), 302–314 (2009)
23. Feamster, N., Balakrishnan, H.: Detecting BGP Configuration Faults with Static Analysis. NSDI (2005)
24. Bush, R., Griffin, T.: Integrity for virtual private routed networks. IEEE INFOCOM 2003 2, 1467–1476 (2003)
25. Al-Shaer, E., Marrero, W., El-Atawy, A., El-Badawi, K.: Network Configuration in A Box: Towards End-to-End Verification of Network Reachability and Security. In: 17th IEEE International Conference on Network Protocols (ICNP 2009), pp. 123–132 (2009)
26. Emerson, E.A.: Temporal and Modal Logic. In: Handbook of Theoretical Computer Science, ch. 16, vol. B, pp. 995–1072. MIT Press (1990)
27. Bryant, R.: Graph-based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers C-35(8), 677–691 (1986)
28. ConfigChecker, `http://www.arc.cdm.depaul.edu/projects/ConfigChecker`
29. McComb, T., Wildman, L.: User guide for SIFA v.1.0. Technical report (2006)
30. Baier, C., Katoen, J.-P.: Principles of Model Checking. The MIT Press (2008)
31. Kotenko, I., Polubelova, O.: Verification of Security Policy Filtering Rules by Model Checking. In: Proceedings of IEEE Fourth International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2011), pp. 706–710 (2011)
32. Holzmann, G.: The Spin Model Checker Primer and Reference Manual. Addison-Wesley (2003)
33. McMillan, K.: The SMV System, `http://www.cs.cmu.edu/_modelcheck/smv.html`
34. Alur, R., Anand, H., Grosu, R., Ivancic, F., et al.: Mocha User Manual. Jmocha Version 2.0, `http://embedded.eecs.berkeley.edu/research/mocha/doc/j-doc/`

Individual Countermeasure Selection Based on the Return On Response Investment Index

Gustavo Gonzalez Granadillo[1], Hervé Débar[1], Grégoire Jacob[1], Chrystel Gaber[2], and Mohammed Achemlal[2]

[1] Telecom Sudparis, SAMOVAR UMR 5157
9 rue Charles Fourier, 91011 EVRY, France
{gustavo.gonzalez_granadillo,herve.debar,
gregoire.jacob}@telecom-sudparis.eu
[2] Orange Labs, Caen, France
{chrsytel.gaber,mohammed.achemlal}@orange.com

Abstract. As the number of attacks, and thus the number of alerts received by Security Information and Event Management Systems (SIEMs) increases, the need for appropriate treatment of these alerts has become essential. The new generation of SIEMs focuses on the response ability to automate the process of selecting and deploying countermeasures. However, current response systems select and deploy security measures without performing a comprehensive impact analysis of attacks and response scenarios. This paper addresses this limitation by proposing a model for the automated selection of optimal security countermeasures. In addition, the paper compares previous mathematical models and studies their limitations, which lead to the creation of a new model that evaluates, ranks and selects optimal countermeasures. The model relies on the optimization of cost sensitive metrics based on the Return On Response Investment (RORI) index. The optimization compares the expected impact of the attacks when doing nothing with the expected impact after applying countermeasures. A case study of a real infrastructure is deployed at the end of the document to show the applicability of the model over a Mobile Money Transfer Service.

Keywords: Impact Analysis, Countermeasure Selection, Risk Mitigation, Return On Response Investment, Mobile Money Transfer Service.

1 Introduction

Current Security Information and Event Management systems (SIEMs) constitute the central platform of modern security operating centres. They gather events from various sensors (intrusion detection systems, anti-virus, firewalls, etc.), correlate these events, and deliver synthetic views for threat handling and security reporting.

Research in SIEM technologies has traditionally focused on providing a comprehensive interpretation of threats, in particular to evaluate their importance

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 156–170, 2012.

and prioritize responses accordingly. However, in many cases, threat responses still require humans to carry out the analysis and decision tasks e.g., understanding the threats, defining the appropriate countermeasures and deploying them. This is a slow and costly process, requiring a high level of expertise, and remaining error-prone nonetheless. Thus, recent research in SIEM technology has focused on the ability to automate the process of selecting and deploying countermeasures.

The authors of [1,2] have proposed automatic response mechanisms, such as the adaptation of security policies, to overcome the limitations of static or manual response. Although these approaches improve the reaction process (making it faster and/or more efficient), they remain limited since these solutions do not analyse the impact of the countermeasures selected to mitigate the attacks. In this paper, we propose a novel and systematic process to select the optimal countermeasure from a pool of candidates, by ranking them based on a trade-off between their efficiency in stopping the attack and their ability to preserve, at the same time, the best service to normal users.

In order to quantitatively analyse the impact of the attack, our model considers two aspects of security policies related to threat responses: firstly, the cumulative long term security policies changes due to previous attacks; and secondly, the fact that security policies may need to be automatically adapted to the current context.

Taking into account previous quantitative models [3–11] this paper proposes a model to select the countermeasure that provides the highest benefit to the organization. It adjusts the proposal made in [10,11] for the use of cost-sensitive metrics to evaluate the impact of each security countermeasure, which allows the system to select the one that provides the maximal RORI index.

The rest of the document is structured as follows: Section 2 introduces the state of the art on impact analysis. Section 3 describes the proposed model to assess countermeasures. Based on the previous assessment, Section 4 explains the process for selecting appropriate countermeasures. Section 5 provides a case study to clearly identify the applicability of the model. Related works are presented in Section 6. Finally, conclusions and perspectives for future work are presented in Section 7.

2 State of the Art on Impact Analysis

Several authors have proposed cost sensitive metrics to balance intrusion damage and response costs, and to guarantee the choice of the most appropriate response without sacrificing the system functionalities. Table 1 describes these models.

The simplest and most used approach for evaluating financial consequences of business investments, decisions and/or actions is the Return On Investment (ROI) metric. ROI compares the benefits versus the costs obtained for a given investment [3,4].

The Return On Attack (ROA) has been defined in [5], as the gain the attacker expects from a successful attack over the losses that he sustains due to the

adoption of security measures by his target. Authors state that the effectiveness of security technology investments could be degraded due to context changes without affecting the ROI index. Additionally, ROI alone is unable to catch the different impacts that solutions have on attackers' behaviours.

The Return On Security Investment (ROSI) metric, has been proposed in [6,7], as a metric that compares the differences between the damages of Information Security incidents (with and without countermeasures) against the cost of the solution. Authors agree that even though the expected damage and mitigation metrics are inaccurate, if the method for determining ROSI produces repeatable and consistent results, the model can be useful for comparing security solutions based on relative values.

More recently, The Return On Response Investement (RORI) has been introduced in [10,11] as a service dependency model for cost sensitive response based on a financial comparison of the response alternatives. The RORI index considers not only response collateral damages but also response effects on intrusions.

Table 1. Summary of Cost Sensitive Models

Models	Return On Investment (ROI) [3,4]	Return On Attack (ROA) [5]	Return On Security Investment (ROSI) [6–9]	Return On Response Investment (RORI) [10,11]
Main Focus	Security Solution Cost (SecCost), Effectiveness	Attack Gain (Att_Gain), Attack Cost (Att_Cost), Losses due to Security (SecLoss)	Security Investment (SecInv), Benefits and Cost	Collateral Damage (CD), Operational Costs (OC) and Response Costs(RC)
Formula	$\frac{Benefit-SecCost}{SecCost}$	$\frac{Att\_Gain}{Att\_Cost+SecLoss}$	$\frac{Returns-SecInv}{SecInv}$	$\frac{Losses-RC-OC}{CD+OC}$
Optimal Solution Characteristics	Highest ROI value Evaluate financial consequences of business investments	Lowest ROA value Evaluate the impact of security solutions based on the attack's behaviour	Highest ROSI value Compare the difference between damages of IT incidents (with and without countermeasures) against the solution cost	Highest RORI value Determine the benefit obtained in a particular threat scenario that applies a given countermeasure
Constraints	- It cannot be used to evaluate the fact of doing nothing - Unable to evaluate the solution's impact due to attacker's behaviour - It does not consider collateral damage nor operational costs	- Difficult to be accurate while predicting attacker's behaviour - It does not consider security solution cost - It cannot be used to evaluate the fact of doing nothing	- It does not consider collateral damage nor operational costs - It cannot be used to evaluate the fact of doing nothing - Unable to evaluate the solution's impact due to attacker's behaviour	- It does not consider attacker's behaviour - It is not defined to evaluate the fact of doing nothing - It is not normalized to the size of the infrastructure

3 Countermeasure Selection Model

Our solution considers the approach proposed by Kheir et al. [10,11], where authors evaluate the Return On Response Investment (RORI) through the formula depicted in Equation 1.

$$RORI = \frac{[ICb - RC] - OC}{CD + OC} \tag{1}$$

Where,

– ICb is the expected intrusion impact in the absence of security measures. It measures the damage cost due to intrusions or attacks.

- RC is the combined impact for both intrusion and response. RC represents the sum of the expected intrusion impacts after a response is enacted and the cost that is added by the selected response.
- OC is the operational cost that includes response set-up and deployment costs such as manpower and over provisioning.
- CD is the response collateral damage which represents the cost that is added by the security measure.

3.1 Constraints

The deployment of the Return-On-Response-Investment (RORI) index into real world scenarios has presented the following shortcomings:

1. The absolute value of parameters such as ICb and RC are difficult to estimate, whereas a ratio of these parameters is easier to determine, which in turn reduces errors of magnitude.
2. The RORI index is not defined when no countermeasure is selected. Since the operational cost (OC) is associated to the security measure, the RORI index will lead to an indetermination when no solution is enacted.
3. The RORI index is not normalized with the size and complexity of the infrastructure.

3.2 Improved RORI

We propose an improvement of the RORI index by taking into account not only the countermeasure cost and its associated risk mitigation, but also the infrastructure value and the expected losses that may occur as a consequence of an intrusion or attack. The improved RORI index handles the choice of applying no countermeasure to compare with the results obtained by the implementation of security solutions (individuals and/or combined countermeasures), and provides a response that is relative to the size of the infrastructure. The improved Return on Response Investment (RORI) index is calculated according to Equation 2.

$$RORI = \frac{(ALE \times RM) - ARC}{ARC + AIV} \times 100 \tag{2}$$

Where,

- ALE is the Annual Loss Expectancy and refers to the impact cost obtained in the absence of security measures. ALE is expressed in currency per year (e.g., $/year) and will depend directly on the attack's severity and likelihood.
- RM refers to the Risk Mitigation level associated to a particular solution. RM takes values between zero and one hundred ($0 \leq RM \leq 100$). In the absence of countermeasures, RM equals 0%.
- ARC is the Annual Response Cost that is incurred by implementing a new security action. ARC = OC+CD from Equation 1. ARC is always greater than or equal to zero ($ARC \geq 0$), and it is expressed in currency per year (e.g., $/year).

– AIV is the Annual Infrastructure Value (e.g., Cost of equipment, Services for regular operations, etc.) that is expected from the system, regardless of the implemented countermeasures. ARC is greater than zero ($AIV > 0$), and it is expressed in currency per year (e.g., $/year).

3.3 Improvements

– The $ICb-RC$ parameters are substituted by $ALE \times RM$ which can be used more easily to evaluate response goodness of single and combined solutions, while reducing error magnitude.
– The AIV parameter also provides a response relative to the size of the infrastructure. AIV is correlated to the Annual Loss Expectancy (ALE) of the system, and allows to compare the RORI results of different systems regardless of their size.
– The introduction of the Annual Infrastructure Value (AIV) parameter handles the case of selecting no countermeasure, which results into a value of zero, meaning that no gain is expected if no solution is implemented.

3.4 Sensitivity Analysis

RORI is a relative index that indicates the percentage of benefit perceived if a given countermeasure is implemented. When analysing the investment in information security, we should not expect an increase in the profits, instead, we should expect a mitigation of the risk to which the organization is exposed.

RORI ranges from $\frac{-ARC}{ARC+AIV}$ (in its lower bound) to $\frac{ALE}{AIV}$ (in its upper bound). A positive RORI means that we expect to diminish the risk up to certain level and therefore it is convenient to apply the security solution. For instance, a RORI of 50% means that we expect to mitigate half of the risk to which the organization is exposed. However, when evaluating the option of doing nothing (no countermeasure is evaluated to react against an attack), we should expect 0% of mitigation.

The worst scenario (the countermeasure cost is higher than the benefits it provides) will have $ALE \times RM << ARC$, therefore $RORI \rightarrow \frac{-ARC}{ARC+AIV}$. The best scenario (perfect mitigation) will have RM=1, ARC=0, therefore $RORI = \frac{ALE}{AIV}$. If the expected benefit is equal to the countermeasure cost, RORI will tend to zero. However, if the expected benefit is lower than the countermeasure cost, RORI will attain a negative value. Only in those cases where the benefit is higher than the cost of implementing a security measure, RORI will attain a positive value.

In order to evaluate the effects on the RORI results, we conducted a series of sensitivity analyses where two variables were changed while the others kept their base case values. The results obtained are described as follows:

– If ARC is orders of magnitude below AIV ($ARC \ll AIV$), then the impact of ARC on the RORI is very weak. In this case, $ARC + AIV \cong AIV$, therefore $RORI \cong \left(\frac{ALE \times RM}{AIV}\right)$. However, if ARC is orders of magnitude above AIV

($ARC \gg AIV$), then the impact of ARC on the RORI index is very strong. In this case, $ARC + AIV \cong ARC$, therefore $RORI \cong \frac{(ALE \times RM) - ARC}{ARC}$.
- If ALE is orders of magnitude below AIV ($ALE \ll AIV$), then ALE negatively impacts the RORI index, since $ALE \times RM \cong 0$, therefore $RORI \cong \frac{-ARC}{ARC+AIV}$. However, if ALE is orders of magnitude above AIV ($ALE \gg AIV$), then the RORI index is positively impacted. In this case, $AIV \cong 0$, therefore $RORI \cong \frac{(ALE \times RM) - ARC}{ARC}$.
- If ALE is orders of magnitude below ARC ($ALE \ll ARC$), then ALE negatively impacts the RORI index, since $ALE \cong 0$, therefore $RORI \cong \frac{-ARC}{ARC+AIV}$. However, if ALE is orders of magnitude above ARC ($ALE \gg ARC$), then the RORI index is positively impacted. In this case, $ARC \cong 0$, therefore $RORI \cong \frac{ALE \times RM}{AIV}$.
- If RM increases compared to the AIV, ALE and ARC values, the RORI index will depend on the magnitude of the ALE metric compared to ARC and AIV. In this case, $ALE \times RM \cong ALE$, therefore $RORI \cong \frac{ALE - ARC}{ARC+AIV}$, making the solution more attractive as the ALE increases.

Table 2 summarizes the results from the sensitivity analysis.

Table 2. RORI Sensitivity Analysis

Parameters	Conclusions
ALE vs AIV	The impact of the ALE parameter over RORI increases as the AIV decreases. The higher the aiv, the less attractive the solution. As a result, a benefit ($ALE \times RM$) that is far greater than the infrastructure value (AIV) is always preferable.
ALE vs RM	The impact of ALE over RORI increases as the RM increases. The higher the ALE and RM values, the more attractive the solution. Thus, the ideal solution should provide the highest benefit to the system.
ALE vs ARC	The impact of ALE over RORI increases as the ARC decreases. The lower the annual response cost, the higher the RORI results. Consequently, a countermeasure that is far less expensive than the benefits it provides is preferable.
RM vs ARC	The impact of ARC over the RORI index decreases as the RM increases. A countermeasure that costs more than the benefits it provides should be discarded. Thus, the ideal solution should have the highest risk mitigation value and the lower response cost.
RM vs AIV	The impact of the AIV over RORI decreases as the RM increases. The higher the RM and the lower the AIV, the more attractive the solution. Consequently, the ideal solution should have the highest risk mitigation and the lowest infrastructure value.
AIV vs ARC	The impact of ARC over RORI increases according to its relative significance compared to the AIV parameter. As a result, an alternative that is far less expensive than the infrastructure value is preferable.

4 Countermeasure Selection Process

The process for selecting optimal countermeasures is performed in two steps: The RORI Calculation and The Countermeasure Evaluation. This section details each part of the model.

4.1 RORI Calculation

The Return On Response Investment metric proposed in Section 3 is used as a quantitative approach to evaluate and rank a set of countermeasures, which allows to select the one that best mitigate the effects of a given attack. The input parameters for the RORI calculation are of two types: fixed parameters (e.g., ALE, AIV), which depend on the intrusion or attack; and variable parameters (e.g., RM, ARC), which depend on the countermeasure.

Fixed Parameters

- The Annual Loss Expectancy (ALE), which refers to the Impact Cost that is produced in the absence of countermeasures. The ALE metric is calculated by multiplying the Single Loss expectancy (SLE) and the Annual Rate of Occurrence (ARO). For this, it is necessary to estimate the severity and likelihood of the attack. The severity of a security incident refers to the impact level produced over an asset. The impact can be measured by loss of system functionality, problems of availability, inability to meet the business goal, monetary losses, etc. The severity ranges from insignificant through to grave according to Lockstep [12], who proposes a corresponding single loss expectancy for each category. The likelihood is an estimation of the frequency or probability that a threat will exploit a given vulnerability. Lockstep [12] proposes six levels of likelihood, from negligible to extreme along with its corresponding numeric values.
- The Annual Infrastructure Value (AIV), which corresponds to the fixed costs that are expected to have on the system regardless of the implemented countermeasure (e.g., equipment purchase, costs of electricity, maintenance, etc). This is a one time investment and remains constant on the evaluation of all the different countermeasures. For this, it is necessary to divide the purchase value into the lifetime of the security equipment and consider depreciations and similar factors, in order to determine the exact cost per year.

Variable Parameters

- The Risk Mitigation (RM), which considers the percentage of reduction of the total incident cost that is given from the implementation of a security measure. Following the Norman's methodology [13], the risk of an attack (R) is determined as the product of its vulnerability (V), likelihood or probability (P), and severity or consequence (C), (i.e., $R = V \times P \times C$). A risk can be mitigated by decreasing the vulnerability, probability, and/or consequence. Therefore, the risk mitigation value (RM) is calculated as the risk reduction percentage that results from the application of a given countermeasure. For instance, if the risk of a given attack before countermeasure is $R_1 = 10 \times 7 \times 7 = 490$ and the resulting risk after the application of a particular countermeasure is $R_2 = 7 \times 6 \times 6 = 252$, then $RM = 100 - (\frac{R_2 \times 100}{R_1}) = 51,43\%$
- The Annual Response Cost (ARC), which refers to the costs associated to a given countermeasure. It includes direct costs (e.g., implementation, consulting and support services during the deployment, maintenance, audits,

analysis, etc) and indirect costs (e.g., consequences that may originate the adoption of a particular countermeasure to a legitimate user).

4.2 Countermeasure Evaluation

The countermeasure evaluation process initiates by selecting the first countermeasure on the list and calculating its RORI index. The obtained RORI is compared with the one by default, as depicted in Figure 1.

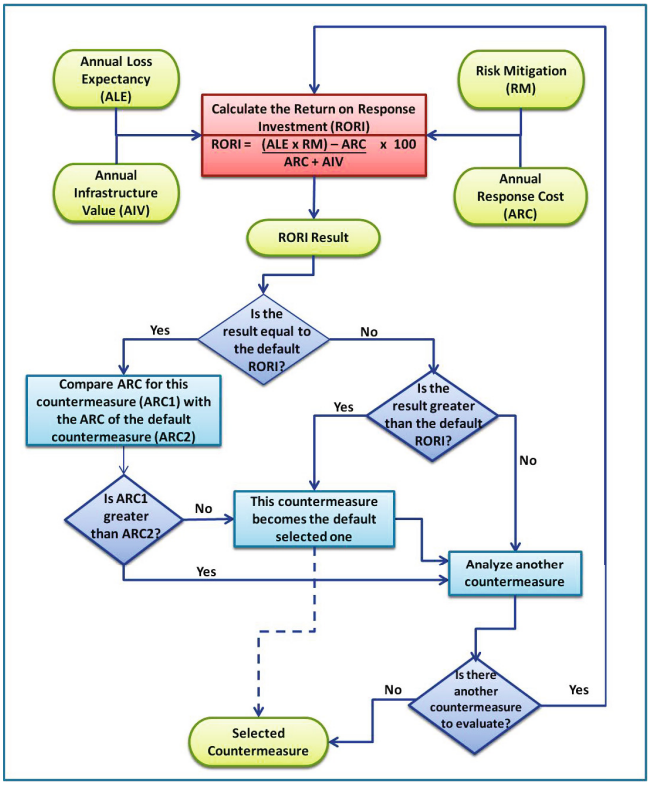

Fig. 1. Countermeasure Evaluation Flowchart

We set up a default RORI value equal to zero (RM and ARC parameters equal 0, since no countermeasure is implemented). If the resulting RORI is different to the one by default, the system checks if the current RORI is greater to the default value, in such a case, the countermeasure becomes the selected one, and it overrides the default RORI value. However, if the resulting RORI value is lower than the one by default, the system checks for another countermeasure to evaluate and the default RORI remains unchanged.

If the resulting RORI is equal to the default value, the system checks for the annual response cost (ARC) and selects the one with the lowest cost value

(it is always preferred to implement a security solution that costs the least and provides the highest benefit). It may happen that, when comparing the costs of two countermeasures, they are exactly the same. In such a case, the system keeps the current RORI value as the default one and checks for another solution to evaluate.

The process is repeated to evaluate the second countermeasure on the list, then the third, and so on, until no countermeasure is left. The system selects the last countermeasure taken as default, since it is the one that provides the highest RORI index.

4.3 Limitations of the RORI-Based Countermeasure Selection

The evaluation and selection of countermeasures depends on the appropriate estimation of the infrastructure value, attack impact and the definition of the security policies needed to mitigate the attack. Such definition should include the costs and benefits associated to a particular security policy in a given attack scenario.

The main limitation of the RORI-based model is the accuracy in the estimation of the different parameters that compose the formula. Estimating the Annual Loss Expectancy of an attack to occur on a given system and the Risk Mitigation level of a particular countermeasure is difficult and requires a considerable effort. An objective estimation of these two factors is rather infeasible, since it requires predictions of an event that has not yet occur.

In addition, the RORI model presented in this paper does not consider interdependence among countermeasures (i.e., how the application of a countermeasure affects the effectiveness of others), nor it discusses the restrictions and/or conflicts that may originate with the implementation of the selected countermeasure (e.g., partially or totally restrictive countermeasures).

Finally, the model limits the action of only one countermeasure over a given attack and does not discuss neither the effects that one or multiple countermeasures may have on several risks nor the effects of applying multiple countermeasures at a time (subject under study).

5 Case Study: Mobile Money Transfer Service (MMTS)

The MMTS case study responds to the needs of improving the security infrastructure of a Mobile Money Transfer Service. The data related to this use case, as well as all numeric values used to estimate costs and benefits have been provided and validated by France Telecom - Orange Group.

5.1 Use Case Description

The MMTS is a system where virtual money, called mMoney is used to carry out several transactions e.g., Bill payments, salary payments, mobile recharge, merchant payments, national and international money transfers, etc. Figure 2 describes this scenario.

The account management server keeps all the information regarding the user's behaviour. The log server registers data that are relevant to analyse abnormal activities such as: failed authentication, request for PIN modification, transaction request, etc. The data warehouse contains historical data about accounts which are useful to analyse customer's behaviour and detect frauds.

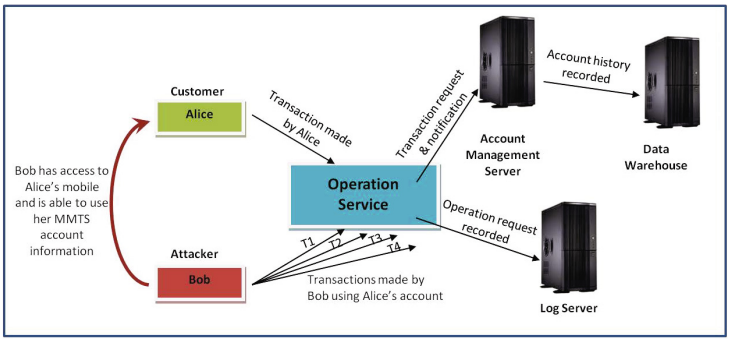

Fig. 2. Mobile Money Transfer Scenario

5.2 Account Takeover Attack

Alice is a user of the Mobile Money Transfer System (MMTS), who only utilizes it two or three times per month in order to pay some bills (e.g., electricity, telephone). Bob (attacker), after a couple of attempts to get authenticated using Alice's credentials, gains access to Alice's MMTS account. He then performs transactions, such as purchasing items and transferring money to a bank account under his control. As a result, the system detects the anomalous behaviour from the logs: For the past two hours, the user Alice, who has always had a regular behaviour (no more than 3 accesses to the system per month), has already used the MMTS several times within a day to carry out some transactions.

An account takeover is a password-based attack that exploits vulnerabilities on the user's side (e.g., social engineering, key-loggers, etc.) and steals the mobile user account to perform transactions in favour of the attacker. An increment on the number of transactions performed in a period of time, or a raise in the amount of money being transferred for a particular user, as compared to the normal behaviour of that user is interpreted as an account takeover attack (an attacker performs some transactions on the MMTS platform using the credentials of a legitimate user).

In agreement with France Telecom - Orange Group, an account takeover attack has an estimated "Minor" severity level[1] (equivalent to 100 €) and a "High" likelihood[1] (once per moth, equivalent to 12). The Annual Loss Expectancy (ALE) for this attack is expected to be 1200 €, and the Annual Infrastructure Value (AIV) is calculated as the value of all the Policy Enforcement Points

[1] The estimated values for the severity and likelihood of an attack may vary from one country to another and depend greatly on the standard of living of each country.

(PEP) that are needed to be deployed in the preliminary phase of the system architecture. Table 3 lists the PEP for this scenario and summarizes information regarding their type, costs and mitigated threats. AIV corresponds to the annual cost of purchasing, licensing, implementing and/or maintaining a security equipment in the MMTS Infrastructure.

Table 3. Security Equipments for a Mobile Money Transfer Service

	PEP	Type	AIV	Threats that mitigate							
				T1	T2	T3	T4	T5	T6	T7	T8
E1	Intrust	HIDS	800€	✓		✓					✓
E2	Tripwire	HIDS	250€	✓		✓					✓
E3	Verisys	HIDS	400€	✓		✓					✓
E4	Snort	NIDS	400€	✓	✓	✓	✓	✓	✓	✓	✓
E5	NetCrunch	Net. Monitoring	1500€	✓	✓	✓	✓	✓	✓	✓	✓
E6	FreeNATS	Net. Monitoring	500€	✓	✓	✓	✓	✓	✓	✓	✓
E7	Comodo	Firewall	300€	✓	✓	✓	✓		✓		✓
E8	Endian	Firewall	150€	✓	✓	✓	✓		✓		✓
E9	Cisco SA 500 series	IPS	1000€	✓	✓	✓	✓	✓	✓	✓	✓
E10	Kaspersky	Antivirus	300€						✓		✓
E11	OS update	OS Hardening	500€				✓		✓		✓
E12	Software Token	Auth. Method	400€	✓	✓	✓	✓	✓	✓	✓	

T1 Trafficking Collection T2 Hiding User Identity T3 Scams
T4 Account Takeover T5 Employee Complicity T6 Denial of Service
T7 Money Creation/Destruction T8 Other threats (e.g. malwares, virus)

From the list of equipments proposed in Table 3, we select 1 NIDS (Snort), 1 network monitor (FreeNATS), 1 Firewall (Comodo), 1 IPS (Cisco SA 500 Series), and a stronger authentication method (software token) as the security solutions to be deployed in a regular system architecture for a Mobile Money Transfer Service, since a combination of all of them provide a wider and more complete coverage of the different threats to which the system is exposed. The AIV is therefore estimated as 2600€(the cost of all the selected solutions).

5.3 Countermeasure Selection for an Account Takeover Attack

In order to react to an intrusion attempt it is necessary to change/update access control policies or implement new mitigation strategies. Table 4 shows the different countermeasures that are proposed to react to an account takeover attack. The second column gives a detailed description of the proposed countermeasures, the third column indicates the Policy Enforcement Point (PEP) that will implement the countermeasure. Columns fourth and fifth of the table detail the risk mitigation percentage (RM) associated to each countermeasure and the annual response cost (ARC) respectively. These figures have been provided by France Telecom - Orange Group. The RORI index shown in the last column is calculated using Equation 2, and it is used to rank the different solutions according to their expected benefit.

From the list of proposed countermeasures, the first alternative (C1) is to accept the risk by doing nothing. This action does not require any modifications

Table 4. Security Measures for an Account Takeover Attack

Countermeasures	Description	PEP	RM	ARC	RORI
C1 Do nothing	Accept the risk	-	0%	0€	0,00%
C2 Deny Transaction	User is unable to perform any transaction	E7	72%	60€	30,34%
C3 Deactivate user account	Temporal deactivation (e.g., 24hrs, 48hrs, 72hrs)	E9	68%	55€	28,66%
C4 Reduce trans. amount	Up to 25€, 50€, 100€	E4	60%	50€	25,77%
C5 Reduce trans. number	1 per day, 3 per day, 5 per day	E4	53%	30€	22,81%
C6 Activate alert mode	Set alert mode and send report	E4	42%	25€	18,25%
C7 Keep account under surveillance	Monitor the user behaviour for 24, 48, 72 hours	E6	42%	40€	17,58%
C8 Activate multi-factor authentication	Request for two or more authentication methods	E12	77%	50€	32,75%
C9 Deactivate mult.trans. requests	The system treats one transaction at a time	E9	64%	20€	28,55%

and therefore the risk remains the same. C1 is totally restrictive and its associated cost is 0, since no countermeasure is implemented. As a result, the RORI index for C1 is 0,00%.

The second and third options suggest to avoid the attack either by denying the transaction (C2), or by deactivating temporarily the user account (C3). As a result, the attack is greatly reduced ($70 - 75\%$). The RORI index for C2 and C3 is 30,34% and 28,66% respectively. Since the RORI index of C2 is greater than the one by default, C2 becomes the default countermeasure.

Alternatives four and five propose to reduce the transaction amount (C4) or to reduce the number of transactions per day (C5), as part of a strategy to prevent attackers from stealing large amount of money from their victims without deactivating the user account. As a result, the attack is mitigated $53 - 60\%$. The RORI index for C4 and C5 is estimated to be 25,77% and 22,81% respectively. The default countermeasure remains unchanged since the RORI index for C4 and C5 is lower than the one by default.

Countermeasures six and seven recommend to activate the alert mode (C6),or to keep the user account under surveillance (C7) e.g., for a period of 24, 48 or 72 hours. Both countermeasures have a risk mitigation value of 42% and a RORI index of 18,25% and 17,58% respectively. The default countermeasure remains unchanged since the RORI results for these two countermeasures are lower than the one by default.

The rest of the countermeasures (C8, C9) recommend to activate additional authentication methods (e.g., two-factor authentication request) and to deactivate multiple transaction requests. By implementing these solutions, the risk is expected to be mitigated $64 - 77\%$. As a result, the RORI index for C8 and C9 is 32,75% and 28,55% respectively.

Since C8 has an index greater than the default RORI, the system proposes to activate the multiple-factor authentication option from the software token PEP in order to guarantee the appropriate authentication of users in the MMTS infrastructure. Alternative C8 becomes therefore the selected countermeasure for an account takeover attack in the MMTS System.

6 Related Work

Most of the existing work in the selection of appropriate countermeasures only concentrates in models that use qualitative analysis or evaluate a single solution for a single attack. Cavusoglu et al. [15], for instance, propose a model to evaluate security investments decisions that uses an attack tree approach based on the game theory. However, the model only considers the implementation of countermeasures in a single attack scenario. Our solution can be adapted to multiple attack scenarios.

Duan and Cleand-Huang consider in [16] heuristic methods and genetic algorithm approaches for the process of selecting a set of countermeasures. However, due to complexity of the search space, the heuristic approach is neither optimal, nor complete. The main drawbacks of the genetic algorithm approach is the difficulty to measure accurately the best portion of each countermeasure to be combined and it uses the Net Present Value (NPV) as the only metric for comparison. Our solution uses ALE instead of NPV, since this latter is not enough to compare among several investments and it is more useful to analyse long period investments. Countermeasures in our model are proposed to be implemented for short period of times (from the moment an intrusion is detected until the system returns to its normal operation).

Furthermore, Neubauer et al. [17] propose the use of efficient safeguard portfolios that are evaluated according to several objectives (e.g., image value, monetary value, accept cost, setup time, etc). However, a moderator is needed to advice during the process, calculations mainly rely on the annual loss expectancy metric, and uncertain outcomes from complex environments are not considered by the tool. Our model does not require a moderator during the evaluation process and utilizes not only the annual loss expectancy but also the operational cost, countermeasure cost and risk mitigation level to evaluate response collateral damages and response effects on attacks.

Bisterilli et al. [18] present a qualitative approach for the selection of security countermeasures using defense trees (an extension of attack trees) and preferences over countermeasures using conditional preference networks (CP-net). However, the conditions for the countermeasure selection are based on expert knowledge, the approach is static and qualitative (no mathematical method is used to evaluate and select the countermeasures), and it does not consider attacks as uncertain variables. Our solution proposes a quantitative model (based on the RORI index) that does not rely on expert knowledge for the selection of appropriate countermeasures.

Zonouz et. al. [19] propose a Response and Recovery Engine (RRE) that uses a tree graph approach to analyse events and select countermeasures based on boolean logic. RRE models a game scenario with two players (an offensive and a defensive), and chooses response actions by solving partially observable competitive Markov decision process that comes from the attack response trees. However, this approach does not consider the benefits and costs associated to a given response action, nor it evaluates quantitatively the different countermeasures to select the one that provides the highest benefit to the organization.

More recently, Bedi et al. [20] describe an approach that uses an algorithm for generating optimal set of countermeasures. However, this approach is proposed to be implemented only during the design phase of software development life cycle; it does not consider unidentified threats; and the impact of countermeasures is not considered in the analysis and selection process. Our solution can be implemented in real time deployments and considers the impact of countermeasures (cost, risk mitigation level) not only during the design phase, but also during the evaluation and selection process.

7 Conclusions and Future Work

In this paper we introduced a quantitative approach to select optimal security countermeasures based on the Return On Response Investment (RORI) index, making it possible to evaluate response collateral damages and response effects on intrusions.

Our solution is split into two steps: the calculation of the Return on Response Investment (RORI) index, which evaluates the expected losses that result for a particular attack versus the benefits that can be obtained if a countermeasure is implemented; and the process of selection and ranking of individual countermeasures. Within the process, the countermeasure with the highest RORI index is selected as the one that provides the highest benefit to the organization. The RORI index takes into account not only the cost and the risk mitigation value associated to a particular solution, but also the losses and operational cost of the infrastructure.

A case study from a Mobile Money Transfer Service is provided to show the applicability of our model and the operations required to evaluate and select the security policies that offer the highest benefit on the system. Future work will focus on managing the conflicts that can be originated from the selected countermeasures (e.g., the implementation of mutually exclusive security policies) and will study the effect of combining two or more security solutions for single and multiple attack scenarios.

Acknowledgements. The work in this paper has been sponsored by the EC Framework Programme as part of the ICT MASSIF project (grant no. 257644).

References

1. Debar, H., Thomas, Y., Cuppens, F., Cuppens-Boulahia, N.: Enabling Automated Threat Response through the Use of Dynamic Security Policy. Journal in Computer Virology 3(3), 195–210 (2007)
2. Riveiro de Azevedo, R., Galvao Dantas, E., Freitas, F., Rodriguez, C., Siqueira de Almeida, M., Campos Veras, W., Santos, R.: An Automatic Ontology-Based Multiagent System for Intrusion Detection in Computing Environments. International Journal for Informatics (IJI) 3(1) (2010)
3. Jeffrey, M.: Return on Investment Analysis for e-Business Projects. In: Bidgoli, H. (ed.) Internet Encyclopedia, 1st edn., vol. 3, pp. 211–236 (2004)

4. Schmidt, M.: Return on Investment (ROI): Meaning and Use. Encyclopedia of Business Terms and Methods (2011), http://www.solutionmatrix.com/return-on-investment.html
5. Cremonini, M., Martini, P.: Evaluating Information Security Investment from Attackers Perspective: the Return-On-Attack (ROA). In: Proceedings of the 4th Workshop on the Economics on Information Security (2005)
6. Brocke, J., Strauch, G., Buddendick, C.: Return on Security Investment - Design Principles of Measurement System Based on Capital Budgeting. In: The 6th International Conference of Information Systems Technology and its Applications (ISTA), vol. 107, pp. 21–32 (2007)
7. Sonnenreich, W., Albanese, J., Stout, B.: Return On Security Investment (ROSI) A Practical Quantitative Model. Journal of Research and Practice in Information Technology 38(1) (2006)
8. Stakhanova, N., Basu, S., Wong, J.: A Cost-Sensitive Model for Preemptive Intrusion Response Systems. In: Proceedings of the 21st International Conference on Advanced Networking and Applications (2007)
9. Kim, D., Lee, T., In, H.: Effective Security Safeguard Selection Process for Return on Security Investment. In: IEEE Asia-Pacific Services Computing Conference (2008)
10. Kheir, N., Cuppens-Boulahia, N., Cuppens, F., Debar, H.: A Service Dependency Model for Cost-Sensitive Intrusion Response. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 626–642. Springer, Heidelberg (2010)
11. Kheir, N.: Response policies and countermeasures: Management of service dependencies and intrusion and reaction impacts, PhD Thesis, Ecole Nationale Superieure des Telecommunications de Bretagne (2010)
12. Lockstep Consulting.: A Guide for Government Agencies Calculating ROSI (2004), http://lockstep.com.au/library/return_on_investment
13. Norman, T.: Risk Analysis and Security Countermeasure Selection. CRC Press, Taylor & Francis Group (2010)
14. Pukkawanna, S., Visoottiviseth, V., Pongpaibool, P.: Lightweight Detection of DoS Attacks. In: 15th International Conference on Networks (ICON), pp. 72–82 (2007)
15. Cavusoglu, H., Mishra, B., Raghunathan, S.: A Model for Evaluating IT Security Investment. Communications of the AMC 47(7), 87–92 (2004)
16. Duan, C., Cleland-Huang, J.: Automated Safeguard Selection Strategies, CTI Research Symposium (2006)
17. Neubauer, T., Stummer, C., Weippl, E.: Workshop-based Multiobjective Security Safeguard Selection. In: First International Conference on Availability, Reliability and Security (ARES), pp. 1–8 (2006)
18. Bistarelli, S., Fioravanti, F., Peretti, P.: Using CP-nets as a guide for countermeasure selection. In: ACM Symposium on Applied Computing, pp. 300–3048 (2007)
19. Zonouz, A., Khurana, H., Sanders, W., Yardley, T.: A Game-Theoretic Intrusion Response and Recovery Engine. In: International Conference on Dependable Systems and Networks (2009)
20. Bedi, P., Gandotra, V., Singhal, A., Narang, H., Sharma, S.: Optimal Countermeasures Identification Method: A New Approach in Secure Software Engineering. European Journal of Scientific Research 55(4), 527–537 (2011)

Security and Reliability Requirements for Advanced Security Event Management

Roland Rieke[1], Luigi Coppolino[2], Andrew Hutchison[3], Elsa Prieto[4], and Chrystel Gaber[5]

[1] Fraunhofer Institute SIT, Darmstadt, Germany
[2] Epsilon S.r.l., Naples, Italy
[3] T-Systems, South Africa
[4] Atos Research & Innovation
[5] Orange Labs - France Telecom

Abstract. This paper addresses security information management in complex application scenarios. Security Information and Event Management (SIEM) systems collect and examine security related events, with the goal of providing a unified view of the monitored systems' security status. While various SIEMs are in production, there is scope to extend the capability and resilience of these systems. The use of SIEM technology in four disparate scenario areas is used in this paper as a catalyst for the development and articulation of Security and Reliability requirements for advanced security event management. The scenarios relate to infrastructure management for a large real-time sporting event, a mobile money payment system, a managed services environment and a cyber-physical dam control system. The diversity of the scenarios enables elaboration of a comprehensive set of Security and Reliability requirements which can be used in the development of future SIEM systems.

Keywords: security requirements, security information and event management, SIEM, architecting trustworthy systems.

1 Introduction

Security information and event management (SIEM) systems provide important security services. They collect and analyse data from different sources, such as sensors, firewalls, routers or servers, and provide decision support based on anticipated impact analysis. This enables timeous response to (or prevention of) attacks as well as impact mitigation by adaptive configuration of countermeasures. However, there are also a number of constraints for current commercial solutions. These constraints include the inability of systems to consider events from a multiple organisations or the ability to provide high degree of trustworthiness or resilience in the event collection environment.

The project MASSIF [3], a large-scale integrating project co-funded by the European Commission, addresses these challenges with respect to four industrial domains: (i) the management of the Olympic Games information technology (IT) infrastructure [12]; (ii) a mobile phone based money transfer service, facing high-level threats such as money laundering; (iii) managed IT outsource services for large distributed enterprises; and (iv) an IT system supporting a critical infrastructure (dam) [4].

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 171–180, 2012.

In undertaking the development of next-generation SIEM concepts and constructs, it became clear that the Security and Reliability of the SIEM itself are critical to the successful deployment of SIEM in a particular environment. With this in mind, we set about analysing each of the mentioned scenarios in some detail, to create an explicit list of Security and Reliability requirements. The intention is that these requirements can be used to guide and assess SIEM development, and ensure that these important attributes are incorporated.

2 Large Scale Scenarios in Four Industrial Domains

In this section, four deployment scenarios for SIEM technology are introduced. The elements of the scenario which can benefit from further SIEM development are also outlined in each case. From the introduction of the scenarios and their unique characteristics, a set of consolidated requirements for a next-generation SIEM can be compiled.

2.1 Scenario 1: SIEM Technologies Used in the Olympic Games

The Olympic Games is one of the largest and most high profile sporting events that takes place, and there is a large technical infrastructure to support many aspects of the games both asynchronously and in real-time. SIEM infrastructure is used with the Olympic Games systems, to protect the games IT infrastructure from any undesired and/or uncontrolled phenomena which could impact any part of the result chain and associated services. The nature of this kind of event presents a big challenge to SIEM infrastructures, for example the next London 2012 games cater for 79 days of competition, 26 sports, 94 venues, 17.000 athletes, 20.000 journalists, 70.000 volunteers, 4.000 IT team members, 900 servers, 1,000 network and security devices and more than 10,000 computers deployed. One of the new challenges will be the amount of data generated from the results systems, representing 30% more than in the Beijing Olympics in order to provide real-time information to fans, commentators and broadcasters world wide. The intensity and complexity of this kind of sporting event presents a big challenge to SIEM infrastructure, mainly, due to two very characteristic features: the number of security event types (about 20,000), and the volume of generated events to be handled (around 11,000,000 alerts per day). However, the most critical aspect that a SIEM system faces in the Olympic Games is that those security events must be processed and reacted upon in real-time.

Advanced SIEM System Contribution. The Olympic Games scenario is valuable to demonstrate the enhancement on scalability, processing enormous amounts of generated data events in real-time. Furthermore the scenario can contribute to validate the cross-layer correlation of events (service, application, infrastructure) from multiple sources.

2.2 Scenario 2: Mobile Money Transfer Service

Use of Mobile Phones to effect payment is a widely used service, particularly in developing markets where banking systems may not be as dense or available as in developed

countries. Characteristics and challenges of authentication, confidentiality, integrity and mobility all have to be considered in this scenario.

From a SIEM perspective, mobile money transfer is an interesting and challenging scenario for the unique attributes that the scenario presents. Indeed, this scenario is quite complex because it requires to analyze past and present data and to extract information from raw events. It is also very sensitive to the performance of detection as the rate of false positives and true negatives should be optimised. Finally, all this should be done while keeping the service scalable and secure. The service allows end users to convert cash to "electronic money" (and vice versa) at merchants, who act as distributors and act as channel users. The electronic money can be used to pay purchases at the merchants' or for bills such as electricity. Furthermore the electronic money can also be transferred between the end users. End users access the service with their mobile phones and distributors can access the service either via mobile phone or directly on the Internet. Both means of access are handled by front-end servers that then access the back-end servers containing the transactions etc.

Advanced SIEM System Contribution. Like any other money transfer service, the service is exposed to the risk of money laundering and other types of fraud. The money laundering risk implies misuse through disguising illegally obtained funds to make them seem legal, and more generally the fraud risk implies any intentional deception made for financial gain. In addition, any money transfer service that has part of its infrastructure exposed via the Internet and/or the end user can access the service using electronic means (a mobile device such as a phone or a pad in this use case), has an increased exposure to fraud, via both attacks against the service infrastructure itself and the abuse of normal service functionality. The objective of including this scenario is to achieve greater protection and transactional integration of SIEM protection through next generation SIEM services. The ultimate intention is to protect the money transfer service against fraud both by detection and application of relevant counter-measures.

2.3 Scenario 3: Managed Enterprise Service Infrastructures

The use of *managed services* by businesses is an increasingly used model, whereby elements of IT and infrastructure are "outsourced" to specialist service providers. In some instances, services are provided by an outsourcer via shared platforms, giving customers economies of scale. In other instances, managed services are performed by a provider on the infrastructure belonging to a customer. Mixed approaches are also possible, and an extrapolation of this can be viewed as occurring when such services are provided in a "cloud based" mode. Provision of Security Information and Event Management services for customers is a valuable complement to the management which an outsourcer or service provider can deliver. The purpose of including a managed enterprise service infrastructure scenario was to consider just such cases: where the services of large enterprises are managed, and a SIEM service is used to collect, inspect and react to large scale security events from member systems and devices.

Advanced SIEM System Contribution. There are a number of limitations of SIEM systems, encountered by managed security service providers, that are not adequately addressed by current SIEM solutions. For this reason, such a SIEM deployment is

interesting to consider when looking at next-generation SIEM requirements. Some of the issues that can be identified in particular are: (i) insufficient resilience of the SIEM infrastructure itself to withstand large scale attacks; (ii) inadequate trustworthiness of source data within the SIEM; and (iii) inadequate disaster recovery capabilities of SIEM systems. Solutions to the limitations that current SIEM systems present will improve the resilience and business continuity capabilities of large companies, through enabling managed service providers to detect and address security events more proactively. It is considered that work on next-generation SIEM systems could address some of the identified problems through the following focus areas:

1. Providing guidelines on the minimum requirements for event data to enable successful event correlation.
2. Providing guidelines on the impact of the unavailability of certain event data on successful event correlation and management.
3. Guaranteeing the trustworthiness of event sources.
4. Improving correlation modelling for better analysis of complex environments (and for better automated correlation processing in complex environments).
5. Improving the resilience and business continuity capabilities for large enterprises.

2.4 Scenario 4: Critical Infrastructure Process Control (Dam)

The features of dam infrastructures are strictly related to the aims they are conceived for. Dams are mostly used for water supply, hydroelectric power generation, irrigation, water activities and wildlife habitat granting. Dams represent fundamental assets for the economy and the safety of a country, such as they are counted among critical infrastructures. So, monitoring of a dam is essential since an accident would have dramatic consequences. The amount of parameters to be monitored to assess the safety of a dam and foresee possible failures or anomalies is enormous, and this huge data flow must be analyzed under real time constraints. Each of these parameters is measured using different sensors, such as inclinometers and tiltmeters, crackmeters, jointmeters, earth pressure cells, turbidimeters, and thermometers. In addition to the above mentioned parameters measured by the sensors, other components are necessary for the full control of dam. Some essential elements are: data collectors, human machine interaction interfaces, data storing units, command and data gateways and signal buses. In other cases there is the need also to integrate different subsystems existing.

Advanced SIEM System Contribution. The current SIEM solutions hardly facilitate the introduction of new technologies to improve the efficiency of the security event detection. At the same time, they usually lack in the capability to support heterogeneous systems and technologies. Introducing SIEMs to jointly manage all different aspects related to the security in the monitoring of a dam can be a very powerful mechanism to increase the overall security of such critical infrastructures. However, currently available SIEMs solutions are focused on the management of digital and information security related events and are designed specifically for this type of applications. This may make complex or even impossible the development of applications targeting security of critical infrastructures in a wider sense. For instance, creating an application capable of correlating network and host events that may indicate a cyber-attack with suspicious

activities detected by the dam surveillance system may greatly improve the security of the whole monitoring process but may introduce some implementation difficulties. SIEMs are not designed to deal with this kind of scenarios and so, encompassing security events coming from different application domains within the same application may be troublesome. In particular, the current technologies usually neglect the possibility to correlate physical and logical events, which can improve the effectiveness of the detection process.

In order to secure the dam control system, today recognized as a critical infrastructure and hence of public interest, regulations must be considered. Indeed, any activity of the dam operators strictly follows well-known rigid procedures. For example, the opening of a gate without alerting the control center is not admitted. Unfortunately, the current SIEM technologies insufficiently exploit regularities characterizing dam systems. In particular, procedures could be encoded in patterns and they could be exploited for detecting anomalies in control system. All this information could contribute to make the security system aware of the context in order to correctly interpret the meaning of some evidences. Introducing such features in a SIEM solution moves the focus of the analysis from a system level view to the business process model of the system.

3 Consolidated Guidelines for Next Generation SIEM

Based on the four scenarios described, and the diverse set of circumstances that they cover between them, a set of consolidated recommendations, to guide the design and development of next generation SIEM platforms, is identified and grouped in five topics.

3.1 Guidelines Concerning Advanced Security Services

Besides issues like dependability, redundancy and fault tolerance, analysis of the four scenarios considered reveals the need for enhanced *security-related* features of future SIEM platforms. These features go beyond what is currently supported by existing solutions. Overall a lack of capability to model incidents at an abstract level is perceived. From the scenarios investigated, and the current SIEM limitations observed, the following guidelines have been identified for next-generation SIEMs with respect to security:

Correlation Across Layers of Security Events. Advanced SIEM systems needs to support enhanced correlation across layers, from network and security devices as well as from the service infrastructure such as correlation of physical and logical event sources. This is due to the variety of systems issuing inputs that can give insights to security only when combined. An example is the off-site monitoring and the on-site management of the dam's configuration.

Multi-level Security Event Modelling. Multi-level security event modelling will enable provision of more holistic solutions to protect the respective infrastructures. The Olympic Games Scenario stipulates that it would be of interest to understand the effects of technical events on the user or process level of the system.

Analysis of Malicious Behaviour Using Attack Graphs. Many of the security issues mentioned in this document originate from complex malicious actions or patterns of actions (e.g., the laundering of money in the mobile money transfer scenario or the misuse case of *Low and Slow* attacks in the Olympic Games infrastructure).

Predictive Security Monitoring. Predictive security monitoring allows to counter negative future actions, proactively. There is a crucial demand for early warning capabilities. Moreover, the limitations with regards to the Managed Enterprise Service point to the fact that dealing with unknown or unpredictable behaviour patterns is not sufficient in current SIEM solutions.

Modeling of the Events and Their Relation to Other, Possibly External, Knowledge. A basic precondition of prediction and simulation as well as of attack analysis is the proper representation of the security requirements and any relevant information about the system as well as any knowledge about the actual and possible behaviour. When reasoning under incomplete information it is not only decisive to properly gather and describe the information available, but it is also required to develop novel methods based on discernibility, probability or plausibility in order to reason about uncertainty.

Securing the Evidence Progressed by the SIEM Components. The misuse case of a sensor compromise, showing that it is vital to be able to trust the information that is received, when using events from sensors like those deployed to monitor the dam or other critical infrastructures.

3.2 Guidelines Concerning Event Processing

Similar to the limitations noted for security, recommendations for *event processing* are also made, based on limitations in current SIEM implementations. The guidelines for a next generation event correlation engine are as follows:

Real-Time. The system must process input data at a high rate and provide meaningful results with soft real-time requirements.

Scalability and Elasticity. The engine should be capable of handling high input rate and should optimize the quantity of resources required based on the actual load. In other words, the system should monitor both input loads and vital parameters, such as CPU utilization, in order to adjust the amount of resources, i.e., provision more resources during peak load times and decommission them during valley load periods.

Handling Streaming and Stored Data. The engine should allow processing and correlation both of streams of events and stored relations (i.e., information stored in a database).

Multiple-Sources. The engine should be able to aggregate, abstract and correlate heterogeneous events from multiple sources at different levels of the system stack.

Pre-defined Correlation Rules and Rule Augmentation Capability. The engine should be shipped with a set of predefined correlation rules to identify well-known attacks. However, it should also support easy and intuitive creation of user-defined rules.

3.3 Guidelines Concerning Advanced SIEM Trustworthiness

Trustworthiness is the ability to provide a service in a way it is expected in terms of safety, security, reliability, availability, and timeliness. The analysis of the input scenarios has resulted in the following guidelines, to improve the general resilience and trustworthiness aspects of a next generation SIEM:

Resilience of the Infrastructure. The infrastructure should be highly resilient under attack, concurrent component failures, and unpredictable network operation conditions.

Security of Event Flows. The event flows should be protected, from the collection points through their distribution, processing and archival.

Protection of the Nodes. The designed mechanisms should offer flexible and incremental solutions for node resilience, providing for seamless deployment of necessary functions and protocols. These mechanisms should take into consideration particular aspects of the infrastructure, such as edge-side and core-side node implementations.

Timeliness of the Infrastructure. The infrastructure should provide for (near) real-time collection, transmission and processing of events, and ensure the corresponding reliable and timeliness generation of alarms and countermeasures when needed. Similarly, features for forensic support should adhere to the following guidelines:

Data Authenticity. Security event data contents, as well as additional/added information related to data origin and destination, must be the reliably stored.

Fault and Intrusion-Tolerant Stable Storage. The stable storage system on which data for forensic use will persist must be tolerant both to faults and to intrusions.

Least Persistence Principle. With respect to sensitive data, only information which is actually needed should be retained to stable storage (much of the data could be processed in real-time and potentially discarded).

Privacy of Forensic Records. Forensic evidence related to security breaches should be made available only to authorized parties.

3.4 Guidelines Concerning Compiler Technologies

In terms of *data acquisition* functionality, it has been noted that next generation SIEM systems should exhibit efficient implementation and/or support for various Features relating to data collection and parsing. Specific guidelines are as follows:

Heterogeneity Support. The data acquisition element must have the ability to deal with a large number of highly heterogeneous data feeds.

High Degree of Adaptability. Seamless integration of new types of security tools/probes should be possible, to improve the capabilities of the SIEM on an ongoing basis.

Peak Handling. The volume of events, to be collected and processed per unit of time, can occasionally increase, resulting in load peaks. The data collection layer should be able to handle such peaks and propagate relevant events to the SIEM core platform without loss of information.

High Degree of Expressiveness. The parsing logic, and related Languages, must allow effective processing of virtually any type of security relevant event.

Support for Fast and Reliable Development. Simple Development and configuration techniques and tools must be available. These will make it possible to implement, deploy, and integrate new parsers and collectors in a relatively short time and at a relatively low cost.

Generality and Platform Independence. The parsing/processing logic (and code) should as far as possible be decoupled from the specific characteristics of the data format and related technologies.

Distributed Processing. Whenever possible (and feasible), the data collection and parsing layer should implement parsing, filtering, and correlation functions at the edges and/or at intermediate nodes, i.e. nodes located along the path to the core SIEM correlation engine.

3.5 Guidelines Concerning Legal Aspects

In terms of *legal* considerations, SIEM systems themselves need to be viewed as data processing entities with consideration being given to issues like data retention, data privacy and so on. From the scenarios considered, the following guidelines in terms of legal aspects have been identified:

Data Retention. Data must be retained for a period of time not more than that necessary to the activities for which they were collected. If the data are required for detection and suppression of crime they can be stored for a longer period of time.

Cross-Border Data Transmission. It must be possible to limit the transmission of data outside of certain borders. It should be possible to process data within such a border. If personal data must be transferred to another country, it must be ensured that the level of data protection in the country of destination is adequate.

Minimum and Appropriate Security Measures. Considering state of the art technology, a minimum (but sufficient) set of measures must be taken to preserve integrity, confidentiality, and availability of personal data. More sensitive data require increased security measures.

Data Minimization and Anonymization. Only data strictly needed for security guarantee must be kept, while unnecessary details must be deleted or made anonymous.

4 Related Work

The development of new security relevant systems requires the integration of a security engineering process in the earliest stages of the development life-cycle. This is specifically important in the development of systems, where security is the enabling technology, as in advanced SIEM systems. There are several common approaches to security requirements engineering that may be taken. An overview of such processes is given in [5] and also in [9]. A comprehensive concept for an overall security requirements engineering process is described in detail in [8]. The authors propose a 9 step approach called SQUARE (Security Quality Engineering Methodology). A similar approach based on the integration of Common Criteria (ISO/IEC 15408) called SREP (Security Requirements Engineering Process) is described in [10]. In [6], different kinds of security requirements are identified and informal guidelines are listed that have proven useful when eliciting concrete security requirements. The author emphasises that there has to be a clear distinction between security requirements and security mechanisms. In [7], Hatebur et al. describe a security engineering process based on security problem

frames and concretised security problem frames. The two kinds of frames constitute patterns for analysing security problems and associated solution approaches. [7] specifically addresses accountability by logging.

Though all of the above mentioned approaches may lead to a sufficient level of security for the designed architecture, there is no obvious means by which they can be compared regarding the security requirements that they fulfil. In this paper, we address the first step in every security engineering procces, namely the identification of artifacts, such as functional descriptions, dependencies and information flows, the identification of use cases and misuse cases, and stakeholders' information on assets, safety and security requirements. Additionally, we consider state-of-the-art information on existing SIEM systems and challenges identified by other work such as the following:

Security information and event management technology provides log management and compliance reporting as well as real-time monitoring and incident management for security events from networks, systems, and applications. A concise overview of current SIEM systems functionalities is presented in [11]. In [1], current threats are identified and advanced monitoring techniques such as file integrity monitoring, database activity monitoring, application monitoring, identity monitoring, and user activity monitoring are discussed. In [2], some challenges with respect to collecting and analyzing a multi-gigabit network stream are outlined. SIEM systems manage security events but are not primarily concerned with the trustworthiness of the event sources. Compared to traditional IT systems, securing SCADA systems (e.g., in the dam scenario) poses unique challenges. In order to understand these challenges and potential dangers, [13] provides a taxonomy of possible cyber attacks – including cyber-induced cyber-physical attacks on SCADA systems.

5 Conclusion and Future Work

This paper has described requirements in terms of security and reliability for advanced security information and event management. The approach used to identify requirements is scenario-driven: scenarios relating to a real-time, high profile sporting event infrastructure; a mobile payment system; an enterprise service provider deployment and a cyber-physical environment has been used as catalyst for requirements identification and elaboration.

Based on the key elements and attributes of each scenario, guidelines for security, event processing, trustworthiness, and compiler technologies in next-generation SIEM systems have been elaborated. To consolidate the approach, a conceptual model showing the progression from business process / application / infrastructure to elements of SIEM design and implementation has been introduced. It is considered to be quite unique and beneficial to have such a comprehensive and rigorous set of scenarios to draw upon, and studying and analysing the scenarios presented provides a sound foundation from which to make recommendations for next-generation SIEM systems.

We cannot necessarily claim that the set of recommendations is "complete", but by developing (and ultimately testing) the proposed items against such a diverse set of scenarios, there is a high probability of addressing a wide range of SIEM requirements. The benefit of multiple scenarios is that associated characteristics which include diverse

requirements including mobility, scalability, real-time processing, potentially hostile device environments and so on. In this light, the security and reliability requirements are considered to be applicable to a wide range of advanced security event management contexts.

Acknowledgements. The authors developed this work in the context of the project MASSIF (ID 257475) being co-funded by the European Commission within FP7.

References

1. Monitoring up the Stack: Adding Value to SIEM. White paper, Securosis L.L.C., Phoenix, AZ (November 2010), https://securosis.com/research/publication/monitoring-up-the-stack-adding-value-to-siem
2. Applied Network Security Analysis: Moving from Data to Information. White paper, Securosis L.L.C., Phoenix, AZ (December 2011), https://securosis.com/research/publication/applied-network-security-analysis-moving-from-data-to-information
3. Project MASSIF website (2012), http://www.massif-project.eu/
4. Coppolino, L., D'Antonio, S., Formicola, V., Romano, L.: Integration of a System for Critical Infrastructure Protection with the OSSIM SIEM Platform: A dam case study. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 199–212. Springer, Heidelberg (2011)
5. Fabian, B., Gürses, S., Heisel, M., Santen, T., Schmidt, H.: A comparison of security requirements engineering methods. Requirements Engineering 15(1), 7–40 (2010)
6. Firesmith, D.: Engineering security requirements. Journal of Object Technology 2(1), 53–68 (2003)
7. Hatebur, D., Heisel, M., Schmidt, H.: Analysis and component-based realization of security requirements. In: Proceedings of the International Conference on Availability, Reliability and Security (AReS), pp. 195–203. IEEE Computer Society Press (2008), http://www.ieee.org/
8. Mead, N.R., Hough, E.D.: Security requirements engineering for software systems: Case studies in support of software engineering education. In: CSEET 2006: Proceedings of the 19th Conference on Software Engineering Education & Training, pp. 149–158. IEEE Computer Society Press, Washington (2006)
9. Mellado, D., Blanco, C., Sánchez, L.E., Fernández-Medina, E.: A systematic review of security requirements engineering. Computer Standards & Interfaces 32(4), 153–165 (2010)
10. Mellado, D., Fernández-Medina, E., Piattini, M.: A common criteria based security requirements engineering process for the development of secure information systems. Comput. Stand. Interfaces 29(2), 244–253 (2007)
11. Nicolett, M., Kavanagh, K.M.: Magic Quadrant for Security Information and Event Management. Gartner Reasearch (May 2010)
12. Prieto, E., Diaz, R., Romano, L., Rieke, R., Achemlal, M.: MASSIF: A promising solution to enhance olympic games IT security. In: International Conference on Global Security, Safety and Sustainability (ICGS3 2011) (2011)
13. Zhu, B., Joseph, A., Sastry, S.: Taxonomy of Cyber Attacks on SCADA Systems. In: Proceedings of CPSCom 2011: The 4th IEEE International Conference on Cyber, Physical and Social Computing, Dalian, China (2011)

Model-Based Security Event Management

Julian Schütte[1], Roland Rieke[2], and Timo Winkelvos[2]

[1] Fraunhofer Institution AISEC, Munich, Germany
[2] Fraunhofer Institute SIT, Darmstadt, Germany

Abstract. With the growing size and complexity of current ICT infrastructures, it becomes increasingly challenging to gain an overview of potential security breaches. Security Information and Event Management systems which aim at collecting, aggregating and processing security-relevant information are therefore on the rise. However, the event model of current systems mostly describes network events and their correlation, but is not linked to a comprehensive security model, including system state, security and compliance requirements, countermeasures, and affected assets. In this paper we introduce a comprehensive semantic model for security event management. Besides the description of security incidents, the model further allows to add conditions over the system state, define countermeasures, and link to external security models.

Keywords: security strategy meta model, security information and event management, complex event processing.

1 Introduction

Today, more and more critical assets are managed by complex ICT infrastructures such as in SCADA systems or heterogeneous and large-scale company networks. Many of these systems are subject to attacks on a daily basis, ranging from mostly harmless drive-by attacks in the form of automated and unsighted scans to targeted insider attacks.

While traditional Security Information and Event Management (SIEM) solutions focus on the mere detection of incidents and usually work at a specific level of abstraction, support for multi-layer correlation and explanation of security implications is scarce. Thus, the relation of any results of these systems to certain security properties or requirements is uncertain. It is hardly possible to derive the consequences of detected incidents on a system scale or process level. Furthermore, it remains a challenge to include information from sensors that go beyond the traditional network and security scanners, especially if these sensors are specific to the domain the system.

Therefore, it is evident that organizations need to broaden their IT monitoring concepts, and incorporate technologies that are designed to look at the application layer and provide detection of application level attacks in near real time [1].

The aim of this work is to enable techniques for interrelating information of different levels of abstraction and of different domains in order to infer more

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 181–190, 2012.

valuable statements about threats in a monitored system. We introduce a modeling approach that facilitates the definition of security probes on different levels of detail, allows to refer to security threats and requirements and enables to integrate a variety of information sources into a thorough information security monitoring.

Current SIEM engines, based on Complex Event Processing (CEP), suffer from mainly three weaknesses, which we try to address in this paper:

First, when incidents are described in a proprietary event processing language, without any semantics linked to the incident definition, it becomes hard for users to understand the actual implications of an incident. Incident definitions are thus more complex, error-prone and harder to maintain.

Second, as the definition of incidents does not follow a formal model, it is not possible to extend it by additional information, such as models of the possible security implications, affected assets, possible remedies, etc.

Third, without such a formal model, it is highly complex to include and correlate additional information from field sensors into the existing incident definitions.

Our approach is therefore to reduce the complexity of security probes by means of an *Security Strategy Meta Model* (SSMM), abstracting from the event processing language. The SSMM allows users to define security monitors at an abstract, less technical layer which is independent from the underlying CEP engine and can be linked to further information, describing possible countermeasures or violated security requirements.

This paper is organized as follows. In Section 2, we reference related work and point out the requirements for the SSMM. In Section 3, we introduce the details of the model and put it into relation to further existing models describing security and infrastructure-specific aspects. Section 4 demonstrates the approach by example of a misuse case and Section 5 concludes the paper and sketches future work.

2 Motivation and Related Work

In this section we first point out what the deficits of existing SIEM solutions are, and derive a list of requirements for an advanced model-driven security event management system.

2.1 State of the Art

The most important types of current threats are identified in [1]; and advanced monitoring techniques such as file integrity monitoring, database activity monitoring, application monitoring, identity monitoring, and user activity monitoring are discussed. In [2], some challenges with respect to collecting and analyzing a multi-gigabit network stream are outlined.

The event and knowledge representations of traditional SIEM solutions show where current limitations of these systems are located, when it comes to a comprehensive analysis of security properties. The existing solutions are very specific and explicitly designed to solve a certain type of problem:

Akab [5] is a SIEM appliance which is mainly focused at monitoring network events. It is based on the proprietary *Akevent* format and stores collected events persistently in a database. Prelude [9] is an open source SIEM framework which relies on the open IDMEF [10] event format. Using the LUA language, developers can write their own correlation modules. The open source SIEM engine OSSIM [4] aims at detecting security events at network (i.e., IP) layer. Consequently, its event format contains attributes like *IP address*, *protocol*, *port number*, and *severity of an incident*. In [8] it is shown how OSSIM can be extended to allow for safety analysis by correlating information which is produced by the security devices adopted in a dam network scenario, with information produced by safety sensor devices. New OSSIM plugins had to be developed and new correlation rules are needed to implement this approach to combined security and safety analysis at runtime. All of these engines have in common that they rely on an event representation syntax, but do not foster a comprehensive event model which links aspects like detection, correlation, reaction, and impact explanation. Moreover, due to the lack of a clear semantics, such event representations cannot serve as a basis for thorough analysis of indicators, which is required to handle potentially huge indicator models. Zabbix, another open source solution focuses mainly at aggregating potentially security-relevant incidents in a common monitoring dashboard and allows users to define simple triggers, e.g. in order to set up notifications.

Among the most mature commercial products are ArcSight ESM and IBM Tivoli Security Information and Event Manager [7]. Their main strength is to relate incidents to compliance catalogs and corporate policies, but the observed events are also predefined by technical attributes such as *source* and *destination host*, *severity*, *user account*, and others [16]. The RSA Archer Threat Monitor maintains a catalog of assets and links it to security-relevant information such as known vulnerabilities, patch levels, etc. Archer itself does however neither collect nor aggregate this information. A common format is the Common Event Format (CEF) [6], also used by ArcSight, for example, but it specifies only the syntax for event representation but does not provide any semantics.

The *Engineering Knowledge Base* (EKB) [17] is an ontology relating sensor values and combining runtime with development time models to analyze industrial automation systems and is used to define SPARQL or SWRL queries over sensor definitions. As we have a similar goal of finding inconsistencies, we believe that an approach like the EKB could help defining which inconsistencies to look for in event streams, and thus, which measurement points might indicate violations of the security requirements. Other approaches of interest to this end are the modeling concepts in [14], where business, application, physical, and technical information is merged and related, as well as concepts to use event-triggered rules for sensing and responding to business situations in [18].

2.2 Requirements

Our aim is to overcome the contextual restrictions of existing solutions with their predefined and closed models and rather provide an extensible model that

comprises all parts of the security monitoring and decision support process: (i) detecting threatening events; (ii) putting them in context of the current system state; (iii) explaining their potential impact with respect to some security- or compliance model; and (iv) taking appropriate actions. Thus, we establish the rationale for the SSMM through a list of requirements that state a set of required properties of the meta-model. In Section 3, we will define the language and processes from which to form a model, which satisfies these requirements.

Requirement 1 (Abstract from event processing languages). *As system operators are not necessarily experts in security monitoring, or SIEM solutions, the model should abstract from the specific event processing languages and vendor specific incident definitions.*

Requirement 2 (Correlation across layers and incidents). *The SIEM engine must be backed by a comprehensive model which allows to correlate incoming events "vertically" and "horizontally".*

Requirement 3 (Inclusion of context information). *Although most current SIEM solutions lack the possibility of correlating alarms with additional context events, it might be necessary in many cases to take additional context information into account.*

Requirement 4 (Model reactions to incidents). *While most SIEM only focus at reporting security incidents, the SSMM should include different ways to handle the incident.*

Requirement 5 (Retro-traceability of security requirements). *It must be possible to automatically link security incident events to a security model that provides additional information about the actual impact of an incident, such as the violated security requirements, concerned assets, and possible countermeasures.*

3 The Security Strategy Meta Model

In this section, we introduce the actual *Security Strategy Meta Model* (SSMM) whose purpose is to describe in a simple and semantically concise way how security incidents should be detected and handled. In the following, we use the term Security Strategy Model for a concrete instance of the SSMM.

An *Security Directive* is the root concept of the SSMM and combines the semantically modeled concepts, which are translated into specific queries and processed by the SIEM engine at runtime. Thus, addressing Requirement 1, a *Security Directive* provides an abstraction from specific event processing languages.

There are two main ways to specify an Security Directive: one is to solely use the structure of the SSMM, but directly formulate queries for specific CEP engines and databases. The other is to describe the Security Directive exclusively using the meta-model. Each Security Directive is structured as follows:

- `on` (`EventStreamProperty`)
- `if` (`Condition`)
- `do` (`Action`)
- `why` (`SecurityPertinence`)

Requirement 2 is addressed by the `on` part, which models security-relevant event patterns by means of an `EventStreamProperty` concept.

Whenever an event pattern is detected, the condition denoted by the `if` part is checked and if it evaluates to true, a security indicident has been detected and requires for a reaction. This allows for inclusion of context and state information, and thereby addresses Requirement 3.

Reactions are modeled by the `do` part (addressing Requirement 4) and refer to an executable `Action`, whereas the model distinguishes between *internal* and *external* actions. Internal actions update the knowledge base, i.e., they can add facts to the domain model or meta data model, as well as they can be used to set system state parameters which can be used in subsequent condition evaluations. External actions, in contrast, refer to loadable plugins which can be used to notify users, write to a database, or take counteractive measures by reconfiguring a firewall, for example.

The `why` part addresses Requirement 5 and refers to an explanation of the incident and may help users to estimate the potential impact and the indicent's relation to the security model. It will be reasonable to link the `why` property either to some compliance catalog (e.g., ISO27004 [13]), or to a formal security model (e.g., the Security Modelling Framework presented in [12]), so as to allow for a quantification of security.

In this paper, we focus mainly on the recognition of security incidents, i.e. the `on` and the `if` property of the Security Directive.

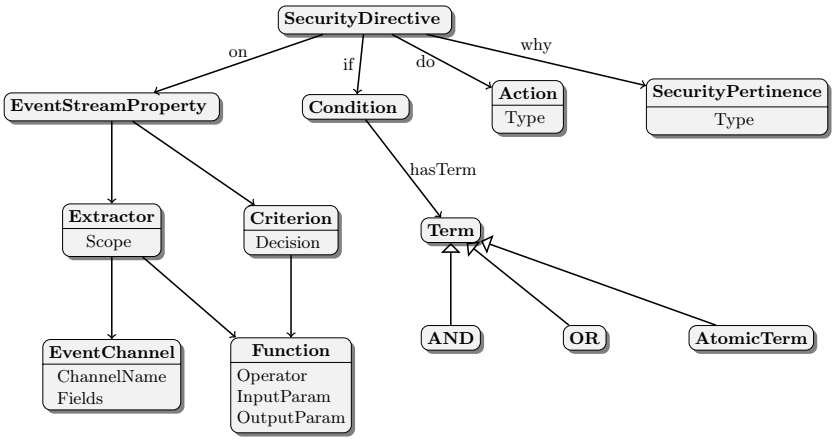

Fig. 1. Structure of a Security Directive

3.1 EventStreamProperty

An EventStreamProperty models patterns of events indicating a security incident. *anomalies*, meaning a property of the behave as expected, or attack *signatures* which are precise event patterns indicating a security incident. An EventStreamProperty comprises the following elements:

- Extractor [1 .. n] Extractors extract attributes from event channels and provide it for further processing.
- Criterion [1] There has to be exactly one Criterion, which triggers the Security Directive. The Criterion specifies how the attributes *extracted* from the event stream should be evaluated.

Extractor. Points to those attributes of an event channel that should be selected from the stream.

- EventChannel [1] denotes the channel from which information is extracted
- Function [0 .. 1] denotes an optional function to be applied to the parameters

Criterion.

- Function [0 .. n] can be applied to to Parameters provided by one or more Extractors (and with that, event channels)
- Makes use of provided values of Parameters
- Decision [1] The boolean parameter which indicates if the Criterion has been positively evaluated. Must be provided by one of the functions above.

EventChannel. Identifies the event channel from which attributes should be extracted. An EventChannel is defined by the following properties:

- ChannelName [1] is the identifier of the channel
- Fields [1 .. n] determine which attributes will be *extracted* from the event channel and provided for further processing
- SchemaName [0 .. 1] identifies the schema of data of the event channel, if any.
- ChannelSensors [0 .. n] describe the sources of an event channel. This additional information might be helpful when porting or reusing the Security Directive.

Function. A function takes the extracted attributes from an event channel as input, applies an operation to them and returns a value. This concept is where the actual processing of event attributes is declared and is thus an essential part of the model. At the moment, Function is predefined by a set of operators which we expect to be used frequently when setting up Security Directives.

- InputParam [0 .. n] specifies input parameters. These can either be static values (e.g., when using threshold functions) or refer to parameters provided by the Extractor.

- `Operator` defines n-ary operators. By logical concatenation, complex operators can be defined and re-used for other Security Directives. At the moment, we support the following operators:
  - Arithmetics: $+, -, \cdot, /, \%\,(mod), ln, log, exp$
  - Comparison: $>, <, >=, <=, ==, \neq$
  - Logical Concatenation: $\wedge, \vee, \neg$
  - Special Functions:
    - $corr(a, b, cor)$: Correlation (with specific *cor* or just $+$ or -, the latter meaning that values of a, b positively or negatively correlate).
    - $avg(parameter, scope)$: Average of `Parameter` over the time frame defined by `Scope`.
    - $max(values)$, $max(field, scope)$, $min(values)$, $min(field, scope)$
    - $anomaly(norm, deviation)$ Defines an anomaly, meaning a deviation from the normal behavior.
- `OutputParam` defines the output. In case of the `Criterion`'s `Function`, this is always a boolean.

`Scope`. A `Scope` defines the window over which aggregate operators like sum, avg, max, or min are applied to event channels. It has the following properties:

- `Type` refers to either number of events, or time in seconds
- `Value` denotes the actual number of events or second.

3.2 `Condition`

A `Condition` allows to match Security Directives only in certain system states and is evaluated whenever a `Criterion` is positively asserted. It refers to a boolean expression over `Terms` and when evaluated, returns a modal decision (`true`/`false`), along with a set of `Parameters`, representing the results of specific `Queries`.

- `hasTerm Term` [1] (`AtomicTerm` | `AND` | `OR`)
  `AND` and `OR` represent the boolean operators and refer to two `Terms` again.

`AtomicTerm`. An AtomicTerm consists of a `Query` and a `Criterion`. The SSMM includes different query types, each comprising the actual query string, the set of parameters which the query provides to the SIEM engine, and a set of query-specific meta information, such as a database name for an SQLQuery, for example. Currently, the SSMM includes the following query types:

- `SQLQuery` Queries an SQL database. Besides the query string, database name, account, and URL have to be provided.
- `AMSECQuery` Queries the Attack Modelling and Security Evaluation Component [15]
- `PSAQuery` Queries the Predictive Security Analyser [19,11]

4 Modeling a Misuse Case

For illustrating the application of the model in a misuse case, we consider a hypothetical but realistic attack in a SCADA system for monitoring a dam infrastructure, based upon the security requirements that were derived in the project MASSIF [3] from the Terni hydroelectric complex, located about 150 Km in the north of Rome: the attacker, one of the workers at the dam system, steals the administrator password for the dam control station. He then uses his own legitimate RFID badge to enter the dam control station room and logs into the control system using the stolen administrator password. With the administration console under his control, he installs a software that intercepts and drops all control messages from the power plant. In the following, he sabotages the discharge sensor so that it would not indicate an increased discharge through the penstocks. Finally, he opens the discharge gates so water flows uncontrolled through the penstocks and the turbine starts to produce energy.

Listing 1.1. Security Directive to monitor the correlation of current and throughput

```
Security Directive Name=ManipulationSensorsSabotage {
:on [ :dischargeCurrentCorrelation
      :hasExtractor [
        :hasEventChannel [ rdf:type :DischargeLevel;
          :hasFields "throughput";
          :hasName "Discharge Level in Penstock"
          :hasChannelSensors :dischargeSens ]
        :hasExtractor [
        :hasEventChannel [ rdf:type :PowerInductionLevel;
          :hasFields "current";
          :hasName "Power Induction Level"
          :hasChannelSensors :currentSens ]
        :hasCriterion [ :hasFunction [:hasOperator :correlate
                                      :hasParam1 "throughput";
                                      :hasParam2 "current";
                                      :outputParam "correlation" ];
                                   :hasFunction [:hasOperator :lt
                                      :hasParam1 "correlation";
                                      :hasParam2 0.3;
                                      :hasOuputParam "threshold" ];
                        :hasDecision "threshold" ] ]

:if [ rdf:type :SQLCondition;
      :hasQuery ="SELECT Role, Employer FROM PhysicalPresenceTable" ;
      :hasCriterion [ :hasFunction [ :hasOperator "==";
                                     :hasParam1 "Role";
                                     :hasParam2 "\"Admin\"" ]];
    :dbInfo [ :ipAdress "192.168.178.54"; :portNumber=31337]  ]

:do  [... left out for the sake of brevity ...]

:why [... left out for the sake of brevity ...]
}
```

Listing 1.2. Generated EPL Query

```
SELECT   sourceIP?,
         avg(cast(traffic?,float)) AS avgTraffic
FROM     SyslogChannel.win:time(30 sec)
HAVING   cast(avgTraffic?,float)>42
```

Due to the compromise of the water flow sensors, the dam control station does not indicate the increased flow through the penstocks. Furthermore, as requests from the power plant have been blocked by the malicious software, requests from the power plant to stop the discharge are ignored and the turbine continues to produce power in an uncontrolled way. This can lead to a situation called *islanding*, in which a part of the electric grid is separated from the rest of the grid, resulting in severe damage of the turbine and the grid infrastructure.

The following listing shows how this attack could be detected by correlating measurements across different layers. The security directive correlates sensor values from the water flow through the penstocks with the produced current at the turbine. As in normal operation, the values should always be positively correlated, the security directive detects the attack described above whenever the correlation falls below a threshold of 0.3.

This scenario mainly served as a guideline for our prototype implementation. We wrote the model as an OWL2 ontology, which is parsed by our prototype engine and compiled into queries for the Esper CEP engine (c.f. Listing 1.2). Whenever an incident is detected, the SIEM prototype links back the detected event pattern to the incident model, so users receive a semantic description of the incident, which they can use as a starting point for a more detailled inspection.

5 Conclusion

In this paper we introduced a model-based approach to the definition of event driven security incident detection and handling. The model supports security monitoring by correlating events from different layers. Detected complex events can be matched against the current system state, where we intend to support different components representing the network infrastructure, as well as components providing attack and vulnerability information, and a predictive security analyser. Some of these components are currently developed within the MASSIF project. Furthermore, the model includes references to actions to be taken when an incident has been detected. In order to put incidents into the context of high-level security requirements, e.g. from compliance catalogs, the model comprises a SecurityPertinence link. We deem the strength of this model-based approach as threefold:

First, the model is comprehensive and brings together all parts of security monitoring which are to date covered by different systems: detection (IDS), reporting (SIEM), handling (like an IRS), and explaining (GRC) of security incidents. So, the model will support an integration of these existing systems into one coherent monitoring solution.

Second, the model abstracts from specific event formats, sensors, or query languages and thereby allows a mapping to different underlying event engines.

Third, representing Security Directives in a semantic model supports a separation of concerns, where a system administrator might provide details on the network infrastructure, the compliance department might provide a list of high-level requirements, and a security officer will combine them in a Security Directive definition, for example.

Our future work aims at extending the existing prototype, integrate it with an IF-MAP server and linking it to specific compliance catalogs, in order to test its practical usefulnes.

Acknowledgments. This work has been developed in the context of the project MASSIF (ID 257475), co-funded by the European Commission within the Seventh Framework Programme.

References

1. Monitoring up the Stack: Adding Value to SIEM. White paper, Securosis L.L.C., Phoenix, AZ (2010)
2. Applied Network Security Analysis: Moving from Data to Information. White paper, Securosis L.L.C., Phoenix, AZ (2011)
3. Project MASSIF website (2012), http://www.massif-project.eu/
4. AlienValult: AlienVault Unified SIEM (2010), http://www.alienvault.com/
5. Araknos: Akab2 (July 2012), http://www.araknos.it/en/prodotti/akab2.html
6. ArcSight Inc.: Common event format: Event interoperability standard (August 2006), http://www.arcsight.com/collateral/CEFstandards.pdf
7. Buecker, A., Amado, J., Druker, D., Lorenz, C., Muehlenbrock, F., Tan, R.: IT Security Compliance Management Design Guide with IBM Tivoli Security Information and Event Manager. IBM Redbooks (July 2010) ISBN 0-7384-3446-9
8. Coppolino, L., D'Antonio, S., Formicola, V., Romano, L.: Integration of a System for Critical Infrastructure Protection with the OSSIM SIEM Platform: A dam case study. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 199–212. Springer, Heidelberg (2011)
9. CS: Prelude SIEM (July 2012), http://www.prelude-technologies.com
10. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765 (Experimental) (March 2007)
11. Eichler, J., Rieke, R.: Model-based Situational Security Analysis. In: Proc. of the 6th Int'l Workshop on Models@run.time at the 14th Int'l Conf. on Model Driven Engineering Languages and Systems (MODELS 2011), Wellington, New Zealand, CEUR Workshop Proceedings, vol. 794, pp. 25–36. IEEE Computer Society (2011)
12. Gürgens, S., Ochsenschläger, P., Rudolph, C.: On a formal framework for security properties. Computer Standards & Interfaces 27, 457–466 (2005)
13. Iec, I.: ISO/IEC 27004:2009 - Information technology - Security techniques - Information security management - Measurement. ISOIEC (2009)
14. Innerhofer-Oberperfler, F., Breu, R.: Using an enterprise architecture for it risk management. In: Proc. of the ISSA Conf. from Insight to Foresight (2006)
15. Kotenko, I., et al.: Analytical attack modeling. Tech. Rep. Deliverable D4.3.1, MASSIF Project (2011)
16. Lieberman Software: Common event format configuration guide (January 2010)
17. Melik-Merkumians, M., Moser, T., Schatten, A., Zoitl, A., Biffl, S.: Knowledge-based runtime failure detection for industrial automation systems. In: Workshop Models@run.time. pp. 108–119. CEUR (2010)
18. Schiefer, J., Rozsnyai, S., Rauscher, C., Saurer, G.: Event-driven rules for sensing and responding to business situations. In: Int'l Conf. on Distributed Event-Based Systems (DEBS), pp. 198–205 (2007)
19. Verissimo, P., et al.: Massif architecture document. Tech. Rep. Deliverable D2.1.1, MASSIF Project (2011)

Using Behavioral Modeling and Customized Normalcy Profiles as Protection against Targeted Cyber-Attacks

Andrey Dolgikh, Tomas Nykodym, Victor Skormin, and Zachary Birnbaum

Binghamton University, Binghamton, NY, USA
{adolgik1,tnykody1,vskormin}@binghamton.edu

Abstract. Targeted cyber-attacks present significant threat to modern computing systems. Modern industrial control systems (SCADA) or military networks are example of high value targets with potentially severe implications in case of successful attack. Anomaly detection can provide solution to targeted attacks as attack is likely to introduce some distortion to observable system activity. Most of the anomaly detection has been done on the level of sequences of system calls and is known to have problems with high false alarm rates. In this paper, we show that better results can be obtained by performing behavioral analysis on higher semantic level. We observe that many critical computer systems serve a specific purpose and are expected to run strictly limited sets of software. We model this behavior by creating customized normalcy profile of this system and evaluate how well does anomaly based detection work in this scenario.

Keywords: Behavior Based IDS, Automatic Signature Generation.

1 Introduction

Modern malware demonstrates features of high precision weapons: information attacks perpetrated by malicious codes can be employed to conduct espionage, and as in the case of the Stuxnet worm, even industrial sabotage [1], [2]. Consequently, Intrusion Detection is a very active area of research that continues evolving as the malware techniques are being improved to overcome existing defenses. However, the most popular malware detection schemes are still dominated by the binary signature-based approach. Although it has many practical advantages, this technology can be evaded by using automatic tools like code packers and metamorphic engines, and leads to a dead end due to exponentially growing database of binary signatures. In addition, it is inherently incapable of addressing targeted, zero-day malware attacks not represented by a binary sample in a database.

Behavioral analysis offers a more promising approach to malware detection since behavioral signatures are more obfuscation resilient than the binary ones. Indeed, changing behavior while preserving the desired (malicious) functions of a program is much harder than changing only the binary structure. More importantly, to achieve its goal, malware usually has to perform some system operations (e.g. registry

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 191–202, 2012.

manipulation). Since system operations can be easily observed and they are difficult to obfuscate or hide, malicious programs are more likely to expose themselves to behavioral detection. Consequently, while database of specific behavioral signatures is still to be utilized, its size and rate of increase are incomparably lower than those in the case of binary signatures. However, the behavioral detector has to be able to distinguish malicious operations from benign ones (i.e. executed by a benign program) which is often difficult. Moreover, maliciousness of an executed functionality can often be determined only by its context or environment. Therefore, the challenge of behavioral detection is in devising a good model of behavior which is descriptive enough to allow for discrimination of benign versus malicious programs and which can be tuned to the target environment.

In principle, there are two kinds of behavior detection mechanisms: misuse detection and anomaly detection. Misuse detection looks for specific behavioral patterns known to be malicious, while the anomaly based approach responds to unusual (unexpected) behavior. The advantage of anomaly based detection is in its ability to protect against previously unseen threats; however, it usually suffers from a high false positive rate. Misuse detection is usually more reliable in terms of detection performance (fewer false positives and often no false negatives) but it has two major drawbacks. First, defining a set of malicious patterns (signatures) is a time consuming and error prone task that calls for periodic updating, similarly to how binary signatures are used today. Second, it cannot detect any malicious code that does not expose known malicious behavior patterns and thus its capabilities to detect a zero day attack are very limited. Consequently, it seems logical to combine both detection mechanisms thus resulting in a highly dependable IDS technology.

In this paper we describe a mechanism capable of automatic discovery of behavioral profiles for computer programs applicable to both malicious and benign behaviors.

First, we discuss the formalization aspects of behavioral signatures representing functionalities, either benign or malicious, in the inclusive and obfuscation resilient form. Then we address the approach enabling the functionality extraction from a Kernel Object Access Graph capturing how kernel objects (objects managed by operating system, e.g. files, processes) are manipulated. Then we discuss the formation of a database containing the functionalities pertaining to the particular network environment that in combination with their frequencies of execution constitutes a customized normalcy profile. The resultant IDS would perform an ongoing task of functionality detection and assess the deviation of the observed network behavior from the earlier established normalcy profile. Finally, the implementation results of the particular components of the described system will be presented.

The described technology is expected to be instrumental in the detection of targeted information attacks against high value targets such as banks, power plants, government installations, etc.

2 Approach

The problem of detecting of unknown behavior can be split in two phases:

During the off-line phase, we observe the stream of the system level events for a time period sufficient to cover the majority of application cases. This accumulated data is used to build a behavioral model of the known behavior of the system. The assembly of the behavioral model can be done off-line with extensive use of computing resources.

During the on-line phase, we match observed stream of events with models of known behavior. If the stream deviates from the behavior predicted by the model we declare the anomaly. In order to be practical, this step should be performed with low overhead.

In this paper we will focus on the process of assembling behavioral model from the continuous stream of the system calls.

This intention is not new, many attempts were made to create models of normal and malicious behavior. Different types of mechanisms: finite automata, context free languages, Markov models etc. were utilized to monitor data dependencies and system call dependencies on order to grasp the complex relationship between system calls and data they operate on. In our view, limited success of these efforts is attributed to the fact that they typically lead to elusive and unstable models which only remotely reflect the behavior of the program. In this paper we introduce a behavior modeling approach operating on the highest level of behavioral semantics, the level where behavior could be directly associated with the specific goals of the software developer.

3 Behavioral Representation

The behavior refers to the actions performed by the program with respect to its environment. The environment of any program in a computer system is controlled and managed by operating system kernel. Windows OS organizes the environment using OS objects: file, memory section, thread, mutex, etc. For a program to sense or modify the environment a request to kernel must be issued. The request to OS is issued by invoking a system call with desired parameters.

For example, system call ZwOpenKey has the following parameters: KeyHandle, DesiredAccess, ObjectAttributes.

KeyHandle is a handle to the opened key. Generally, a handle is a context specific reference/tag to the OS managed object. It allows referencing the same object in a sequence of system calls. DeasiredAccess is the value that determines the requested access to the object. ObjectAttributes is a structure that specifies the object name and other attributes. Unlike handles, the names are system wide object identifiers.

Handles and object names are especially important for observing the behavior since handles and names directly correspond to some OS objects. Additionally, equal handles or names provide clear indication that system calls were issued for the same OS object.

Monitoring system calls along with parameters provides system-wide view on behavior. In order to reason about the behavior and especially about anomalous changes in the behavior we suggest such a model that allows us to capture the normal structure of operations over OS objects.

Formally, the model is a vertex-edge labeled graph which is constructed from the stream of system calls,

$$Gm=(V, E, Fv, Fe), \tag{1}$$

where
V – set of vertices,
E – set of edges,
Fv - mapping from **V** to set of system calls **S**.
Fe - mapping from **E** to set of data links **D**.

It is worth noting that the set of system calls S is small and well known. On the other hand elements of D represent all possible values of parameters of system calls. Therefore the set of data links D is unknown beforehand and very extensive. Fortunately, this does not pose a difficult problem since majority of the important parameters take values from the small subset of D.

The graph Gm can be built from the stream of system calls according to the following rules:

1. Labeled vertex v_s is added to **Gm** for each issued system call s.
2. Labeled edge e_d from v_i to v_j is added when v_i and v_j share the same data d and one of the following:
   (a) v_i has d as the output and v_j takes d as the input
   (b) v_i was registered before v_j

For example, calls S1, S2 in (Figure 1. a) have a common parameter C. In the resulting graph (Figure 1. b) nodes corresponding to calls S1, S2 are connected with the directed edge C. Nodes S2, S3 are connected with an edge labeled C.

The described process transforms the stream of system calls into a stream of graphs. Due to the high volume of system calls these graphs usually grow to unmanageable sizes, primarily due to repetitive/cyclic actions. On the other hand, repetitive occurrences of a single system call or some graph substructure do not provide additional dependency information. Therefore it is beneficial to somehow detect and eliminate repetitive structures (Figure 2).

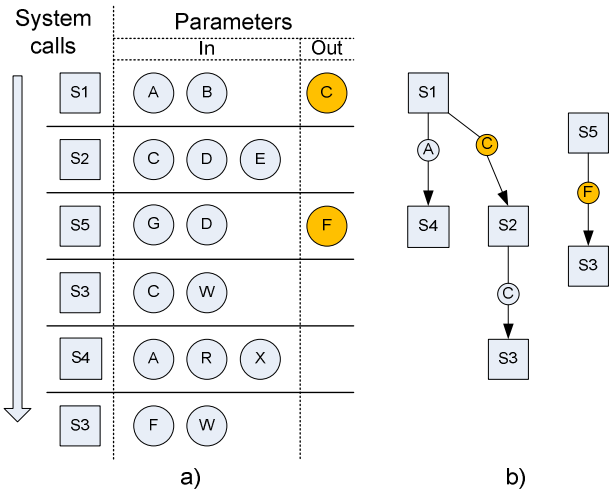

Fig. 1. Links in the graph

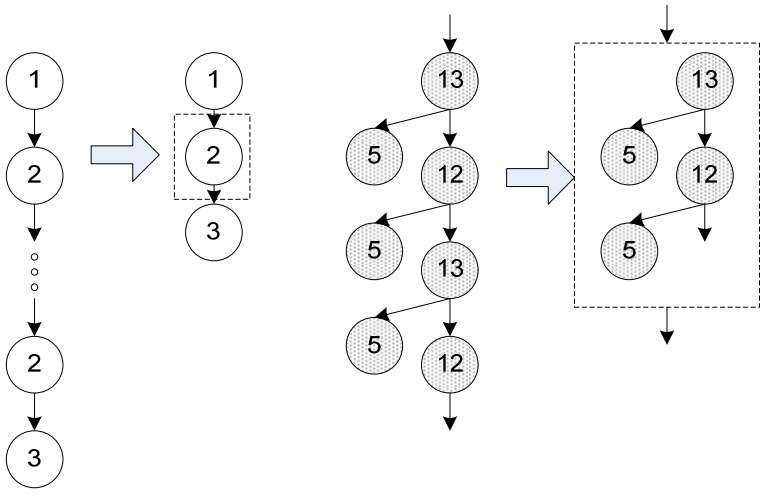

Fig. 2. Graph compression

There are several graph compression (frequent subgraph detection) algorithms suggested in [3, 4]. These algorithms were developed for problems quite different from system call graph compression. The frequent substructures search algorithm from [3] does not scale up to graphs with tens of thousands of nodes. The algorithm described in [4] is more efficient but still too "expensive" for large number of nodes. In addition, the semantics of graph compression described in these papers cannot be directly applied to system call graphs.

Graphs featured in Figure 3 represent overwhelming majority of the system calls graph types observed. Although these graphs are simple in nature, they result in a very heavy performance penalty for general graph compression algorithms. As one may see, they have a very simple structure reflecting repetitive operations over one or two OS objects. Most of huge system call graphs are generated by simple cycles, therefore such traces need to be effectively recognized and eliminated.

For system call graph matching we generally do not care how many times some substructure is repeated. It gives us an opportunity to relax compression accuracy and have irrelevant data disappear. In particular, we replace linear repetitive parts of the graph with one representative component.

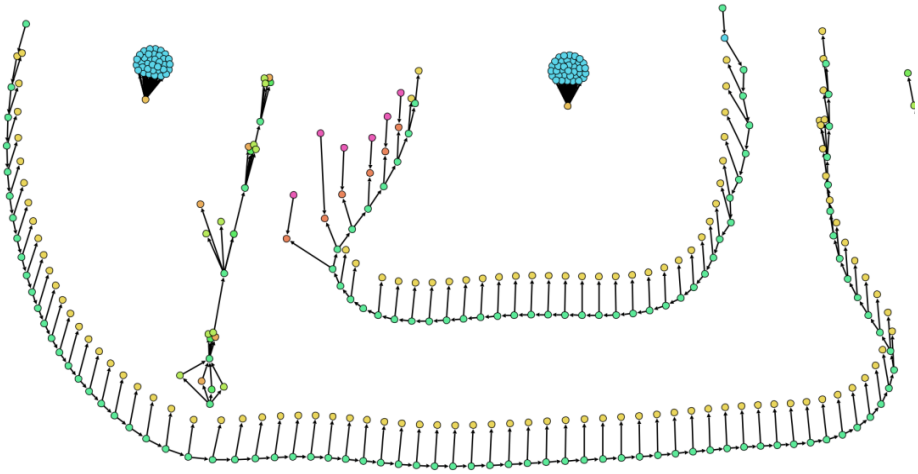

Fig. 3. Typical system call graphs

Lossy graph compression adapted to system call graphs allows for much faster algorithm and better compression of the graph.

4 Algorithm

Our graph compression algorithm is based on Graphitour algorithm which was introduced in [5]. This algorithm proved to be useful in the domain of biological and genetic data processing [6]. It is not the fastest algorithm known to date [7] but it has certain properties that we exploit to our benefit.

We will proceed with brief explanation of the algorithm followed by an example of application of the algorithm to graph compression. The pseudo code for Graphitour algorithm is presented in Table 1.

Table 1. Graphitour graph compression algorithm

```
1  Input: initial graph G
2  Initialize: empty graph grammar, empty edge lexicon;
3  Build edge lexicon: make a single tour of the graph,
          register the type of each edge according to the edge label
          and types of its end nodes; collect type statistics;
4  Loop:
5      Loop through nodes
           Delete all except one leaf nodes of each type
5      Loop through edge types
6          For a sub-graph induced by each edge type solve an
               instance of maximum cardinality matching problem,
               which yields a number and list of edges
               that could be abstracted;
7      Pick an edge type which corresponds to the highest count;
8      Introduce a new hyper-node for the chosen edge type
           into the graph grammar;
9      Loop through edge occurrences of a chosen type in the graph;
10         Substitute an occurrence of the edge type with a hyper-
node;
11         Loop through all edges incident to the end nodes of an
edge;
12             Substitute edges and introduce
               new edge types into edge lexicon;
13 Until no compression possible;
14 Output: compressed graph & induced graph grammar;
```

The major stages of the work of the algorithm are illustrated in Figure 4.

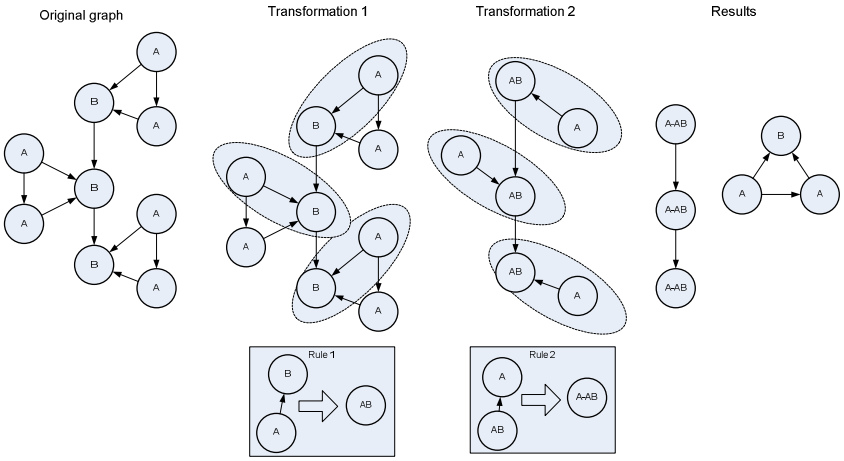

Fig. 4. Application of Graphitour algorithm to the graph with self-similarities

The algorithm first searches for frequent types of edges (lines 5-6) according to max-matching algorithm. The edge A-B induces the largest matching of size 2 (dashed ovals) for the graph featured in Figure 4. The selected edges are contracted and algorithm stores the rule for the applied operation. It could be seen that in later stages previously contracted nodes may participate in new contractions. These successive contractions increase the database of rules which in turn can be unwound into the most frequent patterns occurring in the graph. Unwound rules are featured in the results section of the picture.

Application of Graphitour to typical system call graph is featured in Figure 5. One may notice that after several steps Graphitour collapsed repetitive elements and converted the linear sequence of complex graphs into a trivial sequence of nodes which can be further reduced by removing excessive nodes. Linear parts as featured in Figure 5 stage V can be further analyzed/compressed with application of grammar-based compression as described in [8].

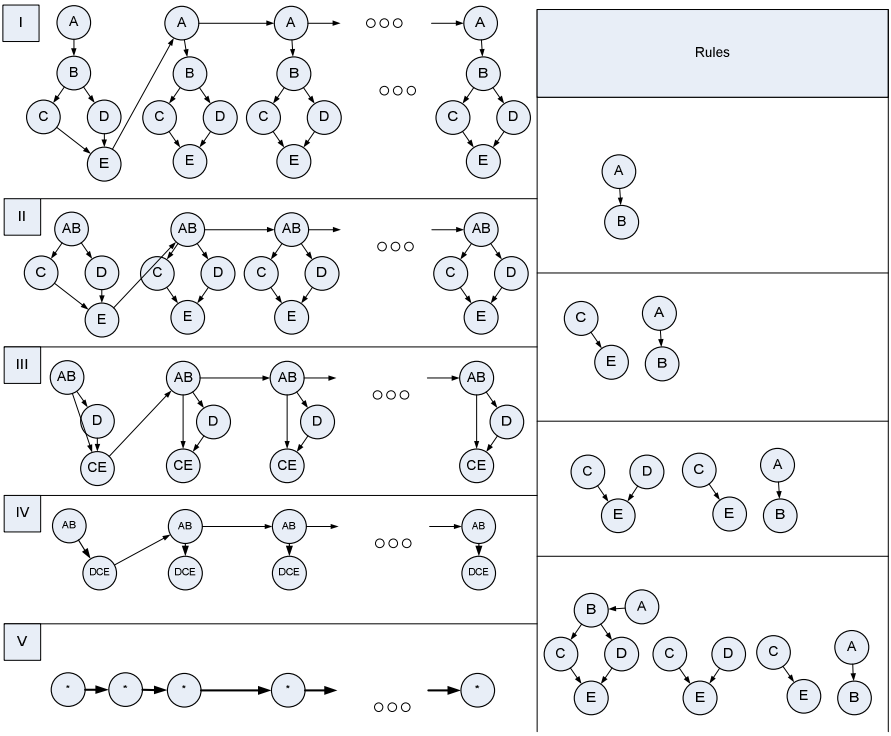

Fig. 5. Progress of Graphitour Algorithm Over System Calls Graph

5 Verification of the Procedure

The described procedure was evaluated on execution traces obtained from several benign and malicious programs running on Windows XP. System call traces were recorded from our driver which intercepted system calls with their arguments by hooking into the SSDT table. Since we wanted to evaluate our approach in general conditions without any prior knowledge about the importance of individual system calls for security, we intercepted all of the calls referenced by the service table, except a few for which we could not find the correct specification of input arguments. We used our driver to obtain execution traces from several malicious and benign applications.

Malicious programs were obtained from the Offensive Computing website [9] and include malware samples of different types and from several families. Benign programs were selected to represent a typical user setup. We joined the obtained samples into three testing traces so that each trace consisted of several malware types and several benign programs.

Since we monitored all of the system calls, the size of execution traces grew rapidly with time, quickly exceeding 10GB for large traces. Therefore we used traces obtained only for a limited amount of time, ranging from 1 minute to 20 minutes in the case of longest execution trace. Currently, the compression was applied to the entire graph that could have over hundred thousand nodes. In the future, some incremental, real time compression schemes could be used allowing the processing of much longer traces. The results could be seen in Table 2.

Table 2. Testing/validation of the functionality extraction procedure

Trace #	Number of system calls	Number of unique graph components	Number of detected functionalities	Number of malicious functionalities detected
1	6927937	1047	341	23
2	3704217	862	307	21
3	20719	217	49	9

Two functionalities extracted from the real system call data by the application of the described procedure can be seen in Figure 6. Component #169 on the left represents typical interaction with Windows registry. Component #171 corresponds to remote thread injection functionality. Such functionality is rarely used by legitimate software. And it is no surprise that it was obtained from the data containing malware execution traces.

6 Normalcy Representation

Rules mined by the Graphitour algorithm over substantially large dataset of system calls representing execution of legitimate software result in a set of functionalities that

can be perceived as signatures of normal behavior. It is important that the described procedure operates without the involvement of human operator, i.e. the functionalities are extracted automatically.

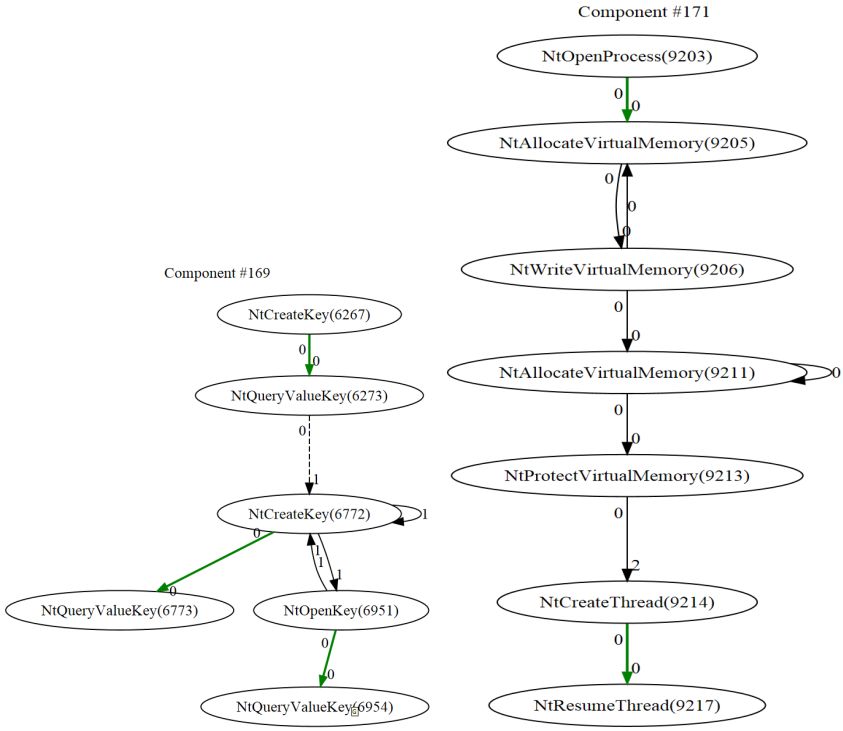

Fig. 6. Functionalities extracted from "real" system call data

The availability of behavior models, i.e. legitimate or malicious functionalities marks the completion of the off-line phase of the modeling effort. The on-line phase calls for the most efficient technology for the functionality detection. We have shown previously that Colored Petri nets (CPN) present a very efficient tool for performing this task [10]. Translation from Graphitour rules (essentially graphs) into CPNs is relatively straightforward. Each node of the rule-graph corresponds to a place in CPN. Incoming edges of the nodes are fused into transitions. Guard expressions of transitions are obtained from edge link types (see Fig. 7).

A set of CPNs obtained from Graphitour mining results that cover all possible legitimate activities for a local network facilitates the anomaly detection. In stable setting this makes our detector more reliable and efficient than SUMMARIZE-MINE mechanism described in [11].

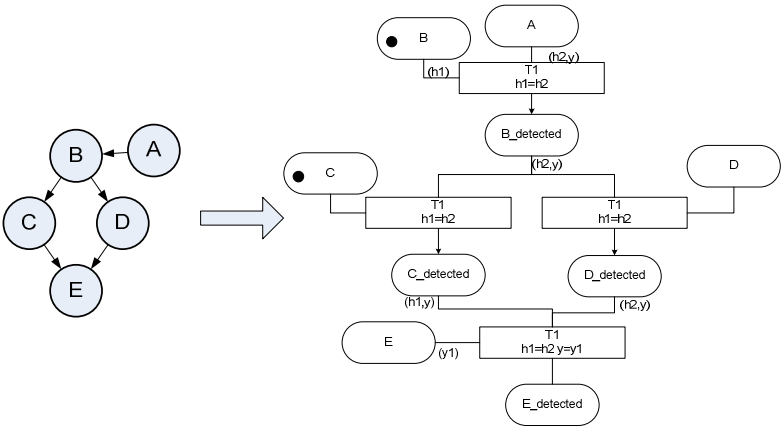

Fig. 7. Translation of Graphitour rules into CPN

The customized normalcy profile is perceived as a set of automatically extracted functionalities, accompanied by the frequencies of their execution. Unlike a public network providing services to a wide community of users, network of a "high value facility" is expected to demonstrate a very rigid set of functionalities and their frequencies. The detection of unseen earlier, not necessarily malicious activity, and/or mere changes in the execution frequencies of the functionalities would indicate an attack. Figure 8 illustrates the described IDS concept.

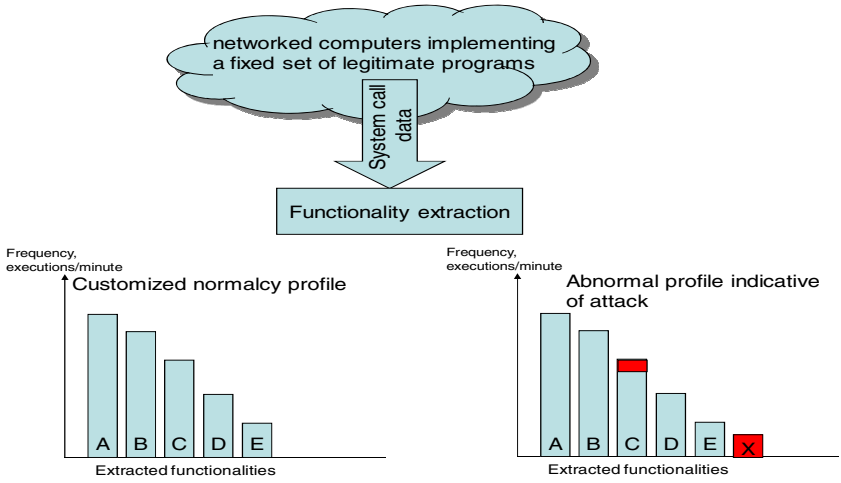

Fig. 8. IDS utilizing a customized normalcy profile

The implementation of the described approach includes the development and periodic updating of the normalcy profile, and the on-going tasks of the functionality extraction, detection of known malicious functionalities, and the anomaly detection in network operation.

Acknowledgement. This research is funded by the Air Force Office of Scientific Research (AFOSR). The authors are grateful to Dr. Robert Herklotz of AFOSR for supporting this effort.

Literature

1. Percoco, N., Ilyas, J.: Malware Freakshow 2010: White paper for Black Hat USA (2010)
2. Falliere, N., Murchu, L., Chien, E.: W32.Stuxnet Dossier: Symantec security response version 1.4 (2011)
3. Cook, D.J., Holder, L.B.: Graph-based data mining. IEEE Intelligent Systems and their Applications 15(2), 32–41 (2000)
4. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
5. Peshkin, L.: Structure induction by lossless graph compression. In: Data Compression Conference, DCC, pp. 53–62 (2007)
6. Hayashida, M., Akutsu, T.: Comparing Biological Networks via Graph Compression. In: Symposium on Optimization and Systems Biology (2009)
7. Choi, Y., Szpankowski, W.: Compression of Graphical Structures: Fundamental Limits, Algorithms, and Experiments. IEEE Transactions on Information Theory (2012)
8. Maruyama, S., Sakamoto, H., Takeda, M.: An Online Algorithm for Lightweight Grammar-Based Compression. Algorithms 5(2), 214–235 (2012)
9. Offensive Computing, `http://offensivecomputing.net/` (accessed, November 2011)
10. Dolgikh, A., Nykodym, T., Skormin, V., Antonakos, J.: Colored Petri nets as the enabling technology in intrusion detection systems. In: Military Communications Conference, MILCOM 2011, pp. 1297–1301 (2011)
11. Chen, C., Lin, C.X., Fredrikson, M., Christodorescu, M., Yan, X.: Mining graph patterns efficiently via randomized summaries. In: Proceedings VLDB Endow, vol. 2(1), pp. 742–753 (2009)

Limitation of Honeypot/Honeynet Databases to Enhance Alert Correlation

Yosra Ben Mustapha, Hervé Débar, and Grégoire Jacob

Telecom Sudparis, SAMOVAR UMR 5157
9 rue Charles Fourier, 91011 EVRY, France
{Yosra.ben_mustapha,Herve.Debar,Gregoire.Jacob}@telecom-sudparis.eu

Abstract. In SIEM environments, security analysts process massive amount of alerts often imprecise. Alert correlation has been designed to efficiently analyze this large volume of alerts. However, a major limitation of existing correlation techniques is that they focus on the local knowledge of alerts and ignore the global view of the threat landscape. In this paper, we introduce an alert enrichment strategy that aims at improving the local domain knowledge about the event with relevant global information about the threat in order to enhance the security event correlation process.

Today, the most prominent sources of information about the global threat landscape are the large honeypot/honeynet infrastructures which allow us to gather more in-depth insights on the modus operandi of attackers by looking at the threat dynamics. In this paper, we explore four honeypot databases that collect information about malware propagation and security information about web-based server profile. We evaluate the use of these databases to correlate local alerts with global knowledge. Our experiments show that the information stored in current honeypot databases suffers from several limitations related to: the interaction level of honeypots that influences their coverage and their analysis of the attacker's activities, collection of raw data which may include imprecise or voluminous information, the lack of standardization in the information representation which hinder cross-references between different databases, the lack of documentation describing the available information.

1 Introduction

Security Information and Event Management (SIEM) systems provide security analysts with large amount of heterogeneous alerts. Managing and analyzing such tremendous number of alerts is a challenging task for security administrator, in particular, because these events often lack precise and concise information [1].

Alert Correlation approaches have been developed to improve the accuracy of alerts and attack understanding [2]. In addition, it provides a better description of the observed threat phenomena. Existing correlation techniques such as [1–9] apply to local alert datasets. The view over the local alert dataset is limited by functional and structural boundaries of the monitored system. Local correlation

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 203–217, 2012.

does not provide methods to determine if the alerts, locally detected, are part of a more global threat phenomena.

In this context, the honeypot technology is a valuable instrumentation technique to automatically collect and learn information about server-based exploits and global threat phenomena. Recent work, [10–12], has shown the usefulness of gathering experimental data to model and better understand the threats due to attackers. The deployment of honeypots in several locations of the IP space has underlined the fact that different blocks of addresses are attacked differently. It is thus extremely important to have in-depth information about these threats in order to study the causality links with local detected attacks.

In this paper, we describe a novel cross-view correlation approach capable of analyzing causality relationships between local alerts detected in the monitored system level and the global threat phenomena observed by honeypot sensor deployment. This approach is beneficial to reassess the attack severity and to re-evaluate the attack impact. In fact, analyzing information about the global threat phenomena attributed to the locally detected alerts apprise us of severity of the global threat phenomena, its propagation strategy, its capabilities, etc. Our approach takes advantage of the data collected by four honeypot databases to enrich our knowledge about alerts. We explore and evaluate these databases in order to enhance the detection report in the alert and the knowledge about the global threat landscape and correlate it with local reported alerts. The two first databases contain information about Internet malware activity obtained through the deployment of distributed honeypot sensors. The second and third databases provide information about web-related threats by analysing security evolution of suspicious web-based servers and domains.

Our experiments show that honeypot databases, in particular those we explore, suffer from several limitation in the context of alert correlation which will be detailed in Section 4.

2 State of the Art

2.1 Honeypots and Honeynet

A *honeypot*, as defined in [13] and [14], consists in an environment where vulnerabilities have been deliberately introduced in order to observe attacks and intrusions and to facilitate in-depth analysis of attackers strategies. It provides its operators with intelligence about threats and network exploits.
Honeypots sensors interact with attackers in different ways, classified according to three types of interaction level as listed in [15]:

1. *low-interaction honeypot:* It emulates the presence of different network services on a host rather than a complete system.
2. *high-interaction honeypot:* Unlike low-interaction honeypot, a high-interaction honeypot emulates a complete real system. It usually runs a full implementation of an Operating System and installed applications. It is able to ensure a real environment with which the attacker will interact. This technique gathers

Table 1. Honeypot interaction levels

Interaction level	Advantages	Disadvantages	Examples	Collected Information
Low-interaction honeypot	– simple implementation – easy to use and maintain – low risk of penetration	– emulate specific services – lower interaction performance – no real interaction – limited scope of attacker's activity – detection of known attacks – detectable by advanced attackers – capture only activity that directly interact with them	Specter, HoneyD	– limited to the level of emulation – time and date of the attack – protocol – source and destination IP address – source and destination ports
High-interaction honeypot	– full implementation of an Operating System – real interaction with attacker	– complex implementation and maintaining – time-consuming to design – increased risk – capture only activity that directly interact with them	Honeynet	– valuable information about the attacker's behaviour – possible information about zero-day attacks
Medium-interaction honeypot	– simulated services are more complicated – low risk of penetration – better interaction performance	– do not emulate a complete real Operating System – capture only activity that directly interact with them	nepenthes, mwcollect, honeytrap	– information about more complex attacks

more details about the modus-operandi of attackers. *Honeynet* is an example of high-interaction honeypot.

3. *medium-interaction honeypot:* It is a more sophisticated honeypot than low-interaction one but it is still not emulating a complete operating system like high-interaction honeypots. It emulated more complicated services.

In Table 1, we summarize the pros and cons of these honeypot technologies and we give examples of each category and more details about the information collected by them.

A common drawback of honeypots is that their field of view is limited and they only gather attacker's activity which interacts with them.

A recent evolution of honeypot research field is proposed in [16] and integrates different tools such as ScriptGen [17], Argos [18] and Nepenthes [19]. In [16], authors propose a distributed system of honeypots in order to gather more details about attacker's activity againt several victim machines and networks.

In [11], authors demonstrate how honeypot databases offer significant data to study malware propagation and to get a deeper understanding of their evolution.

2.2 Alert Correlation Techniques

Alert Correlation is usually defined as an approach that aims at discovering various relationships between individual alerts. The main categories of Alert Correlation techniques are similarity-based, knowledge-based divided into scenario-based and rule-based approaches, and model-based correlation approaches.

***Similarity-based*:** this technique aim at clustering alerts which have the closest similarity values between alert attributes. Most considered attributes are source and destination IP address and ports as well as timestamp. Distance-based function and probabilistic function, as proposed in [6], are computed to evaluate similarities between alerts. Unless these functions are useful to cluster similar alerts which are most likely linked to a common root cause, they are not useful to evaluate to causality links between alerts.

***Knowledge-based*:** this technique is based on a priori knowledge on malicious activities and attacks scenarios and exploits. It requires not only expertise rules and heuristics to correlate heterogeneous alerts but also a consistent information related to alerts. We distinguish between two main sub-categories of knowledge-based techniques:

- *Scenario-based*: a single alert reflects often to an elementary step of an attack scenario. Consequently, various attack scenario templates are required to correlate those alerts. Modelling language such as LAMBDA (Language to Model a Database for Detection of Attacks), [20] and AdeLe (Attack Description Language) have been used for attack scenarios specification. Furthermore, attack scenarios recognition techniques were developed like statistical Granger Causality Test, chronicles formalism and other machine learning techniques to enlarge the set of scenarios.
- *Rule-based*: known as prerequisites and consequences, this technique is designed to correlate alerts following their causality relationships. It is a specific technique that serves for scenario-based alert correlation. But, it does not require a prior knowledge of attack scenarios. Rule-based alert correlation approaches take advantage of the JIGSAW attack description language [4] to model the capabilities of an attack and its steps. These latter are constructed once the capabilities are satisfied. Hence, if some alerts are missed, the attack steps reconstruction remains incomplete. This limitation was addressed by the MIRADOR correlation method which does not require full satisfaction of all capabilities, [5].

These aforementioned alert correlation techniques are essentially based on the information contained in the alert itself and on static databases including information about vulnerabilities such as the Common Vulnerabilities and Exposure CVE database, National Vulnerability Database, Open Source Vulnerability Database, etc.

***Model-based*:** this approach aims at supporting alert correlation at analysing security alerts with a view on the relevant context. M2D2, [7], was the first attempt to offer a formal representation of sensor capabilities in order to decide whether an alert was false positive or not. This was enhanced by the proposed data model M4D4 [8,9], which covers contextual and topology information. This model is a shared model that is developed in order to process in a cooperative way while correlating alerts. It represents the different relationships among system entities and components to facilitate the correlation process by a cooperative analysis of heterogeneous information.

3 Proposed Approach: Cross-View Alert Correlation

As described in 2.2, within traditional alert correlation particularly knowledge-based approaches, security administrator deal with a limited view of raw alerts environment and concentrate on the analysis of the victim-side knowledge related to these alerts. In fact, alerts are sometimes incomplete and do not include

sufficient knowledge about the global attack sequence. Linking local alerts to global phenomena in the threat landscape makes this knowledge available. Honeypot sensors interact with attackers following different level as mentioned in 2.1 to learn more about their propagation strategies, their source characterization, etc. We explore this information (refer to section 4.2) about threat landscape and propose a cross-view alert correlation.

Our novel approach aims at automatically linking local alerts to a more global threat phenomenon by enriching the set of locally observed alerts. It also ensures an effective attack impact re-evaluation by analyzing the knowledge about the attacker capabilities and the global threat phenomena which is attributed to the local detected alerts. Cross-View correlation support also response decision process in selecting a more appropriate countermeasure. In fact, understanding the attackers' modus-operandi and the global threat phenomena that affects our monitored network is necessary to identify better mitigation strategy.

3.1 Information Sources

The proposed cross-view correlation performs on heterogeneous security-related information gathered from two different views: local view and global view.

The local view includes analyzers such as Intrusion Detection Systems (IDS), Firewalls, etc, are deployed to report traces of malicious activity affecting the network and other security related information. For instance, IDSs monitor the activity of the network for the occurrence of malicious activities and generate alerts triggered by their signatures. From the alerts, we can retrieve the timestamp of the malicious activity, the source and target IP address, used port, etc.

The global view includes honeypot data are widely deployed to provide researchers with in-depth information about the *hacking community* and cyber threats. As we described in 2.1, the primary goal of a honeypot is the study of the attacker behaviour while interacting with the sensor. The data collected through honeypot sensor includes information about malware behaviour, propagation vectors used by malware, the propagation strategy, relationships between exploits, attacker's location, source and destination characterization, origin and relevance of zero-day attacks, etc [15,16].

3.2 Alert Enrichement Process

In the context of cross-view alert correlation, we enrich our local knowledge about alerts with information about the global threat landscape. Hence, it is essential to define an alert enrichment strategy that improve the alert-related knowledge, especially with appropriate external information related to the occurrence of the exploit. Figure 1 describes an overview of the alert enrichment process. As shown in figure 1, we first *collect* information respecting appropriate *enrichment features* that are detailed later. For instance, based on the originating source of the observed alert, we collect global knowledge about the tracked server and evaluate its security profile and evolution over time. After the enrichment is performed, we *categorize* the collected information following the defined

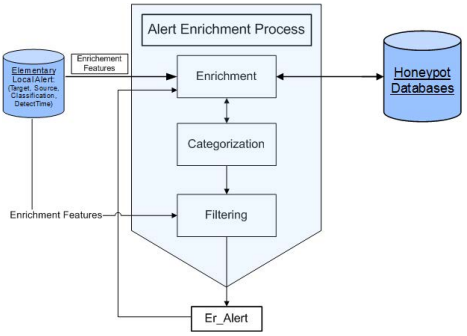

Fig. 1. Alert Enrichment Process

categories presented later and propose a *filtering* process which is composed of three types of filters: Temporal, Semantic and configuration filters. Each filter perform on a corresponding category of information.

The result of the enrichment process is an *Er_alert* which append the information reported basically in the elementary alert with additional information about the security state of the source, threats reported on it which is able to perform tracked attack and in-depth description about these exploits. We then analyse the relationship that may exist between the threat which has been reported on the specific server and the locally reported alerts. This analysis require sometimes a bigger picture about data collected in the first step of enrichment.

The *Er_alert* is then processed to request for more general information based on a generalization strategy of the enrichment feature. For instance, to analyze the environment of the originating source of the local threat, it is possible to request information about its localization, the class of its IP address (class A, B or C).

Local Cross-View Enrichment Features. It is essential to analyze local alert features that must be considered in the enrichment process since we manipulate information from two different views: the local view and the global view.

At the local victim-side, Intrusion Detection Systems (IDS)s monitor the activity of the network for the occurrence of malicious activities and generate alerts of detected malicious activity including elementary information about the infection. Following the Intrusion Detection Message Exchange Format (IDMEF) [21], the alert is composed of several aggregate classes. Source, DetectTime and Classification classes are convenient features that must be considered first when querying honeypot datasets. These features construct a primary bridge to more accurate global information about the detected threat. Hereafter, we detail the information gained by analyzing collected objects when applying these features.

Source class includes the IP address of the originating source of the detected threat. As mentioned in paragraph 2.1, honeypot databases log information related to the source IP address of malicious activity. Thus, alert's source characterization will be considered during the enrichment process.

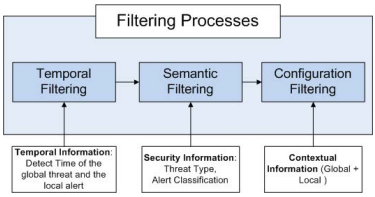

Fig. 2. Filtering Processes

DetectTime represent the time when the attack was detected by the local analyzer. In honeypot databases, captured activities are usually timestamped and allow us to analyze the security evolution of the alert's originating source within a specific time window.
Classification represent the alert semantic. For instance, Intrusion Detection Systems categorize alerts respecting a set of alert classification which inform us about the alert's type. This latter may be linked to the threat type detected by honeypot sensors.

Information Categorization. The aforementioned enrichment features allow us to gather huge amount of related information which sometimes does not enhance our knowledge about the alert. Thus, we propose to categorize the collected information aiming at simplifying the filtering processes.

We define three major categories:

Security Information: This category includes security-related information such as md5, type of the threat, security states of the originating sources, exploited vulnerabilities by the attacker, etc.
Temporal Information: When detecting a malicious activity, honeypots' logged information is timestamped. Honeypot datasets' objects includes temporal attributes about the detection and analysis time.
Contextual Information: This category includes generic type of information: spatial information, whois information, generic information. This set of attributes allows us to build a more general picture about the environment of the object being analyzed.

Filtering Process. The objective of the enrichment process is to increase the accuracy of the knowledge related to the alert. To fulfil this need, we set up 3 filters (as shown in figure 2). The objective of these filters is to eliminate data which is less likely to link the local alert to the corresponding global threat phenomena.
Temporal Filtering: Honeypot sensors track the activity of attackers and the evolution of the threat landscape referring to a time settings. Honeypot datasets include a timeline of events for each tracked source. This timeline is composed of different timespans defined by the first and last time (resp. t_{first_seen} and t_{last_seen}) of the observed activity generated from tracked source. t_{first_seen} and t_{last_seen} are timestamps that are of the order of seconds. We denote by

$T\_th_{IPaddr} = [t_{first\_seen}, t_{last\_seen}]$ the timespan of an observed threat on a specific source. Within the defined cross-view correlation approach, it is important to operate on timespans that are close to the *DetectTime* of the local alert. Obviously, it is not significant to attribute local alerts to a global threat which no longer exists. Therefore, we use sliding windows to avoid investigating the correlation of local alerts with old reported global threats. We denote by $\alpha = DetectTime - t_{last\_seen}$ the delay that exist between the last time of the last activity observed on a source and the detection time of the local alert. We expect α to be of the order of a day. The considered sliding window must satisfy this condition.

Semantic Filtering: As local alerts are usually classified following the attack's objective, global threat phenomena is also assigned to specific type of attack activity. By setting up a semantic filtering, we eliminate non-relevant global threat phenomenon relatively to the classification of the local alert in order to keep only those threats that are highly likely to have a causality relationship with the local alert. The observed global threat is classified respecting a high level description of the threat defined by a *threat_class*. This classification is extremely helpful to characterize threats and the modus operandi of attackers behind them. Honeypot datasets use their own set of threat classes to distinguish between detected global threats. The semantic filtering is then able to immediately compare the threat class against the alert type if the set of threat classes is based on the same semantic of local alerts classification. Even though, the semantic filtering has to consider intermediate cross-references in order to map these different sets of threat classes and alert types.

Configuration Filtering: Cross-view correlation must take into account the contextual knowledge of the monitored network. This knowledge is composed of topological and cartographic data. It represents hosts' characteristics, their interconnections, software products, their vulnerabilities, etc. As mentioned in [9], comparing the affected configuration of a vulnerability with the actual set of products of a given host is valuable for alert correlation. In the context of cross-view correlation, it is meaningful to compare global threat capabilities against the weaknesses of the monitored system and network.

4 Experimental Results and Analysis

4.1 Local Information Source

Our analysis is conducted on real world alerts generated by a snort v2.8 NIDS sensor running on our University Network for 4 months. The University Network is composed of hundreds of machines. For the experiment, we use the latest version of the signature rule-sets available at the time of the experiment procedure which began on January 2012.

During these 4 months, snort generated 183170 alerts originating by 2499 unique IP address source. We summarize these alerts in Table 2, sorted by their classification.

Table 2. Classification of local generated alerts

Classification	Number of alerts	Total Number of unique IP sources	Number of filtered alerts	Number of filtered unique IP sources
attempted-recon	156338	2198	870 (0,5%)	132 (6%)
attempted-dos	1	1	1 (100%)	1 (100%)
attempted-user	1540	28	1540 (100%)	28 (100%)
misc-activity	839	174	23 (3%)	21 (12%)
trojan-activity	3	3	3 (100%)	3 (100%)
bad-unknown	22573	34	51 (0,2%)	9 (26,5%)
unclassified	1876	61	1876 (100%)	61 (100%)
Global	183170	2499	4364 (2,5%)	255 (10%)

Table 3. Summary of reported signatures, their vulnerabilities, protocols and ports

Classification	Signature	CVE Reference	Protocol and Port	N. of alerts
attempted-recon	SNMP request tcp	CVE-2002-0012/0014	TCP:80	3
	SNMP AgentX/tcp request	CVE-2002-0013	TCP:80, 31337	2
	SCAN FIN/SCAN SYN FIN	–	TCP	860
attempted-dos	DoS Teardrop attack	CVE-1999-0015	UDP	1
attempted-user	MS-SQL probe response overflow	CVE-2003-0903	UDP:55989, 56538, 41376, 36845, 64439, 4974	1335
	Web-Client Windows Media Player directory traversal via Content-Disposition	CVE-2003-0228	TCP:80	1
misc-activity	ICMP PING CyberKit 2.2 Windows	–	ICMP	20
	BAD-TRAFFIC tcp port 0 traffic	–	TCP:0	3
trojan-activity	BACKDOOR typot trojan traffic	–	TCP:44086	3
bad-unknown	ICMP Source Quench	–	ICMP	50
	DNS SPOOF query response with TTL of 1 min, and no authority	–	UDP:53	1

Several alerts have been filtered. In fact, in order to increase the performance of the enrichment process, we eliminate several alerts since snort IDS is known by its high rate of false positive. Hereafter, we consider classifications for which alerts have been filtered:

Attempted reconnaissance alerts are usually preliminary intrusion steps aiming at collecting information about the network. Most of these alerts are, in general [22], alerts for normal network activity. In fact, such alerts are more significant if they are part of a global attack sequence. Therefore, we filter alerts whose IP source address is not present in other alerts having different signature. More than 99% of attempt-recon alerts have been filtered (ref. Table 2).

Misc-activity and Bad-unknown alerts include large number of ICMP alerts which can be considered as a Usual False Positive referring to [23]. These alerts are usually generated due to misconfigured hosts, topology of the network, normal activity of network services, etc. As shown in Table 2, more than 99% of misc-activity and bad-unknown alerts have been filtered.

In the last line of the Table 2, we compute the number of alerts generated by our Snort sensor and the number of corresponding unique IP source address. We then aggregate the total number of filtered alerts, with its average referring to the totality of alerts. Finally, we give the number of corresponding filtered IP source address that will be considered during our experiments.

Table 3 shows a more detailed information about the exploited vulnerabilities and the protocol involved in the detected malicious activity. Vulnerabilities are identified by their CVE reference, [24]. Moreover, we represent most important port numbers employed by attackers to infect the monitored network.

4.2 Experiment Honeypot Databases

We explore four honeypot databases which include information about malware characterization and security profile of suspicious web-based servers.

SGNET v1, [16], is a distributed honeypot deployment which benefits from different tools, namely ScriptGen [17], Argos [18] and Nepenthes [19] aiming at gathering proper view on Internet attacks and malware. It collects information about the malware propagation strategies as well as information providing as better understanding about global threat landscape. Recently, an enhanced work of SGNET has been developed and called SGNET v2.

HARMUR v1, the Historical ARchive of Malicious URLs, [25], is a security dataset that aims at exploring the dynamics of the security and contextual information associated to web-related threats. HARMUR extract several security information tracked web-based servers where hosted suspicious domains. It is possible to retrieve threats that was reported on the tracked server. Like SGNET, HARMUR's developers improve several functionalities of HARMUR and they developed the HARMUR v2.

These honeypot databases has been developed initially within the WOMBAT project. A public API called WAPI (Wombat API), [26], has been developed in order to query information from advanced honeypot databases. Information collected in these datasets is object oriented. The specification of the WAPI protocol relies on four different concepts: objects, attributes,methods and references. Aggregation of these concepts offer information on a security object (e.g. an IP address) that is generated by a set of different datasets (SGNET honeypots, HARMUR web servers, ...).

4.3 Experimental Results and Evaluation

As explained in section 3, in our experiments, we perform the enrichment process based on elementary characterization of the alert such as source address or port. For instance, based on the IP address of the alert's source, alert enrichment process allows us to gather more details about potential root causes. Additional features would be considered within the enrichment process such as the hash of potentially detected malwares.

During our experiments, we check if the originating source reported in our set of alerts has been analyzed by one of the honeypot sensors. We analyze the security evolution of alert's source and evaluate and quantify the threat phenomena which infects the originating source of the local detected alert.

Table 4 summarizes statistical results computed during the alert enrichment process. In this table, we represent the number of alert's IP source address having corresponding server object analyzed by one of the honeypot sensor. Indeed, the primary step of the enrichment process is concentrated on the identification of the set of alerts that are systematically enriched.

Based on the result shown in Table 4, we conclude that a high number of alerts' IP source address have not been analyzed by experiment honeypot sensors. Thus, we conclude that several alerts will not be automatically enriched.

Table 4. Experimental Results using alert's IP source address

Classification	Number of filtered IP source address	SGNET v1	SGNET v2	HARMUR v1	HARMUR v2
attempted-recon	132	–	–	**1** (SNMP AgentX/TCP)	–
attempted-dos	1	–	–	–	–
attempted-user	28	–	–	–	–
misc-activity	21	–	–	–	–
trojan-activity	3	–	–	–	–
bad-unknown	9	**1** (DNS SPOOF)	–	–	**1** (DNS SPOOF)
unclassified	61	–	–	**1** (tcp portscan) & **1** (tcp portscan,tcp portsweep)	**1** (tcp portscan)

Table 5. Analysis of identified IP source address objects from Table 4

Signature involved	SNMP AgentX/TCP	DNS SPOOF		tcp portscan	tcp portscan & tcp portsweep	tcp portscan
Number of alerts	1	1		4	203 & 103	1
Honeypot Dataset	HARMUR v1	SGNET v1	HARMUR v2	HARMUR v1	HARMUR v1	HARMUR v2
Temporal Information:						
first_seen, last_seen attributes	unfilled	unfilled	unfilled	unfilled	unfilled	unfilled
Security Information:						
Number of hosted Domains	41	–	1	1	2	1
Security current color of hosted domains	28 green, 8 gray, 4 orange & 1 red	–	1 gray	1 green	2 orange	1 red
Number of corresponding threat objects	25	no threat	–	no threat	4	1
Threats rating	14 unknown & 11 red	–	–	–	4 unknown	–
Type of the threat	Virus, Browser Exploit, Generic google safebrowsing	–	–	–	Virus	–
Number of satisfied content reference	missing	–	missing	missing	missing	missing
number of filled help attribute	13	–	–	–	4	–
information about affected systems	Windows: 98, 95, XP, Me, NT, 2003, 2000	–	–	–	Windows XP, Vista, NT, Server 2003, 2000	–

Moreover, the examples in the experiment honeypot datasets which are linked to the alerts are difficult to explore in the context of our cross-view alert correlation. In Table 5, we detail the information gathered from each IP source address object found in the datasets.

Enrichment: As shown in Table 4, the enrichment process operates on a small set of originating sources. 5 IP source addresses out of 255 IP source addresses has been identified in the experiment honeypot datasets. Due to this reduced number of identified IP source addresses, the performance of the enrichment process are highly reduced. Knowledge about the originating source of the local alerts is not entirely enriched.

Categorization: Based on the sources identified in Table 4, we analyze the information gathered from each corresponding source object. Experiment honeyot datasets set temporal attributes for each object that are composed of last_seen, first_seen. Moreover, security information includes information about the threat reported in each source, their types, their rating, a link to a more detailed description about the threat, the security color of the hosted domains in the source (e.g. red color means that the domain is infected). In the case of the source identified in SGNET v1, all the attributes and references are missing.

Filtering: Temporal information is the principle input for temporal filtering. Normally, this information is of the order of seconds which is a sufficient granularity in the context of our alert enrichment process. Due to the missing of such information in the results, it does not allow the filtering process to be completely performed. A part from this, threat type values (e.g. Virus, Browser

Exploit, Generic google safebrowsing) listed in Table 5 do not offer a standard representation of the threat type which avoid the semantic filtering to be automatically executed. During our experiments, we identify several limitations to deeply explore these honeypot datasets. Hereafter, we point out these limitations on the deployment of such honeypot databases to enrich the alert knowledge in the context of cross-view alert correlation.

4.4 Interpretations

Four major limitations have been identified during the experiments. Some limitations are related to the characteristics of a honeypot sensor implementation. In addition, we identify other limitations which are related to the design of a honeypot datasets data representation.

Coverage Limitation. When requesting the corresponding object of a specific originating source of a local alerts, around 95% of locally detected alerts do not have corresponding originating source reference in the explored databases (as deduced from Table 4). The main cause of this result is explained by the interaction level of the honeypot sensors. Despite honeypot sensors are able to interact with attackers and emulate network protocols, they would be unable to cover a large variety of activities like a real interaction between the attacker and the system. For instance, Web-Client Windows Media Player directory traversal via Content-Disposition attacks (cf. CVE from Table 3) require a high-interaction level with the attacker and this is not always ensured. Such result makes challenging the correlation and enrichment process since we are then conducted to set a generalization methodology. In fact, it is possible to gather information about the network IP address range of an alert's source.

Unfilled Attributes and References Limitation. During our enrichment process, we identify the problem of unfilled attributes of datasets' objects which does not guarantee the automatic reasoning and the continuity of the collection process. For instance, an important number of threats have unfilled attributes such as the help attribute which contain details about the threat type, the vulnerabilities that can be exploited, etc. We conduct a statistical evaluation on 73699 threat objects from HARMUR datasets and we observe that over than 85% of these objects are not referenced to a content object. This latter includes in-depth information related to the threat. This pitfall prevent us essentially from applying semantic and configuration filters described in 3.2. Moreover, from Table 5, due to the absence of temporal attributes, our temporal filtering is not capable to accomplish the filtering process.

Lack of Standard Representation of Data. During our experiments, we observe that information gathered about threats lacks a proper and standard representation. Unfortunately, this limitation makes difficult qualitative evaluation of the exploits and semantic filter. Within HARMUR threats object, each object has a generic threat type as phishing page, virus, browser exploit, etc.

Such threat types do not provide clear understanding and in-depth details about the exploit and construct an obstacle for semantic filtering. For instance, even if honyepot datasets report a threat in a specific server, neither the threat type representation nor the help attribute, allow us to identify if a causality relationship exist between the locally detected alert and this global threat phenomena.

Cross-References Limitation. Since honeypot databases lack of cooperation and coordination between them, it was a tedious task to request in-depth information of mawlare infection reported in a specific web-based server from SGNET that includes information about the malware characterization and behaviour. For instance, it would be interesting to gather in-depth information about a threat reported in HARMUR by requesting SGNET. In the context of our approach of cross-view alert correlation and enrichment, it is required that we take advantage of a complete global view of the threat phenomena such as the infected servers, their security evolution over time, characterization and specificity of exploits.

5 Conclusions and Perspectives

This paper introduces an application of honeypot databases to enhance alert correlation techniques. We expose a cross-view alert correlation that aims at considering external security information collected through the deployment of honeypot sensors in order to enrich the local knowledge of detected intrusions with in-depth details about the security profile of the originating source, exploits to which our network is exposed to, etc. We then analyze and explain experimental results and limitations encountered when dealing with the explored honeypot databases to ensure the cross-view alert correlation. Although honeypot technologies are proved to be a valuable mean to analyze the threat ecosystem, our experiments demonstrate several limitations of the deployment of current honeypot databases to improve alert correlation. For instance, the lack of precise information and standard representation do not ensure correlation analysis and automatic reasoning. This pitfall can be alleviated by conceiving a unified and standardized framework for honeypot data storage and representation. Assigning standard reference such as CVE to observed exploits and threats could be of a great interest while exploring honeypot databases. These suggestions would ensure that cross-view alert correlation is executed. It would also make the navigation and the cross-references between these different honeypot databases easier.

Honeypot techniques are still evolutionary and can be ameliorated to cover additional security analysis application. In the context of our cross-view alert correlation, it is important to ensure concise and complete information that offer to the security analyst a good understanding of threat landscape ecosystem to efficiently identify causality relationships between the local detected alerts and observed threat phenomena in the global view.

Acknowledgement. The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257495, "Visual Analytic Representation of Large Datasets for Enhancing Network Security (VIS-SENSE)".

References

1. Ren, H., Stakhanova, N., Ghorbani, A.A.: An Online Adaptive Approach to Alert Correlation. In: Kreibich, C., Jahnke, M. (eds.) DIMVA 2010. LNCS, vol. 6201, pp. 153–172. Springer, Heidelberg (2010)
2. Chifflier, P., Tricaud, S.: Intrusion detection systems correlation: a weapon of mass investigation (2008)
3. Cheung, S., Lindqvist, U., Fong, M.W.: Modeling multistep cyber attacks for scenario recognition. In: DISCEX (1). IEEE Computer Society (2003)
4. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. In: Proceedings of New Security Paradigms Workshop, pp. 31–38. ACM Press (2000)
5. Cuppens, F.: Managing alerts in a multi-intrusion detection environment. In: Computer Security Applications Conference (December 2001)
6. Valdes, A., Skinner, K.: Probabilistic Alert Correlation. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 54–68. Springer, Heidelberg (2001)
7. Morin, B., Mé, L., Debar, H., Ducassé, M.: M2D2: A Formal Data Model for IDS Alert Correlation. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 115–137. Springer, Heidelberg (2002)
8. Morin, B., Debar, H.: Correlation of Intrusion Symptoms: An Application of Chronicles. In: Vigna, G., Kruegel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 94–112. Springer, Heidelberg (2003)
9. Morin, B., Mé, L., Debar, H., Duccassé, M.: M4D4: a Logical Framework to Support Alert Correlation in Intrusion Detection. Information Fusion 10, 285–299 (2009)
10. Comparetti, P.M., Maggi, F.: Using WOMBAT APIs on Real-World Tasks. In: The second WOMBAT Workshop, pp. 67–81 (2009)
11. Leita, C., Bayer, U., Kirda, E.: Exploiting diverse observation perspectives to get insights on the malware landscape. In: 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2010 (June 2010)
12. Li, L., Sun, H., Zhang, Z.: The research and design of honeypot system applied in the lan security. In: 2nd International Conference on Software Engineering and Service Science (ICSESS). IEEE (2011)
13. Dagon, D., Qin, X., Gu, G., Lee, W., Grizzard, J.B., Levine, J.G., Owen, H.L.: HoneyStat: Local Worm Detection Using Honeypots. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 39–58. Springer, Heidelberg (2004)
14. Pouget, F., Dacier, M.: Honeypot-based Forensics. In: AusCERT Asia Pacific Information Technology Security Conference (2004)
15. Mokube, I., Adams, M.: Honeypots: Concepts, Approaches, and Challenges. In: ACMSE 2007,Winston-Salem, North Carolina, USA (March 2007)
16. Leita, C., Dacier, M.: SGNET: a worldwide deployable framework to support the analysis of malware threat models. In: 7th European Dependable Computing Conference, EDCC 2008 (May 2008)

17. Leita, C., Dacier, M., Massicotte, F.: Automatic Handling of Protocol Dependencies and Reaction to 0-Day Attacks with ScriptGen Based Honeypots. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 185–205. Springer, Heidelberg (2006)
18. Portokalidis, G., Slowinska, A., Bos, H.: Argos: an emulator for fingerprinting zero-day attacks. In: ACM Sigops EuroSys (2006)
19. Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.C.: The Nepenthes Platform: An Efficient Approach to Collect Malware. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 165–184. Springer, Heidelberg (2006)
20. Cuppens, F., Ortalo, R.: LAMBDA: A Language to Model a Database for Detection of Attacks. LNCS (2000)
21. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format (IDMEF), http://tools.ietf.org/pdf/rfc4765.pdf
22. Remi-Omosowon, O.B.: Statistical analysis of snort alerts (2009)
23. Spathoulas, G.P., Katsikas, S.K.: Reducing false positives in intrusion detection systems. Computer & Security 29, 35–44 (2010)
24. Common vulnerabilities and exposures, http://cve.mitre.org/cve/
25. Leita, C., Cova, M.: HARMUR: Storing and analyzing historic data on malicious domains. In: Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (2011)
26. Zanero, S., Comparetti, P.M.: The WOMBAT API: querying a global network of advanced honeypots. In: BlackHat DC (2010)

Stochastic Model of Interaction between Botnets and Distributed Computer Defense Systems

Dmitry P. Zegzhda and Tatiana V. Stepanova

Saint-Petersburg State Polytechnical University
{dmitry,stepanova}@ibks.ftk.spbstu.ru

Abstract. Nowadays one of the main means for computer attack organization are botnets. One of botnets' goals is to break computer defense system and the goal of defense system is to neutralize botnets, staying resistant to its targeted attacks. There is lack of efficiency evaluation methods and models, which allow to compare how sustainable is defense system to targeted botnet attacks and vice versa. Proposed model allows to predict the result of interaction between botnet and defense system and can be used as base for building efficient distributed defense system, capable of protecting itself from botnet attacks.

Keywords: botnet, distributed defense system, efficiency evaluation, sustainability, random graph.

1 Introduction

Malware authors tend to organize networks of malware agents, which work together; such networks are called botnets. Modern botnets have sophisticated organization, decentralized or hybrid architecture, utilize random graph, small world or scale free topology [1]. These properties allow botnets to be highly resistant against neutralization techniques, used by defense systems. Botnets also use methods (in addition to botnets' payload), which aim at breaking defense system itself, thus helping botnets in distribution and performing malicious actions. To protect themselves from such attacks, defense systems also turn to distributed architecture. So, we can talk about nets of cooperative defense agents (NCDA) and nets of cooperative malware agents (NCMA), which represent similar threats to each other. To describe both of these net types, term NCxA will be used. All threats could be divided into three groups as follows:

1. Threats to confidentiality:
   (a) detection the source of control commands;
   (b) detection the source of new nodes creation;
   (c) detection the recipient of the result;
   (d) tracing internal data flow;
   (e) nodes detection;
   (f) disclosure of information about nodes in the net;

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 218–225, 2012.

(g) key information disclosure;
(h) transport protocol disclosure.

2. Threats to availability:
   (a) system control loss;
   (b) communication channel noise;
   (c) communication channel block;
   (d) violation of communication between agents.
3. Threats to integrity;
   (a) net control loss (control still can be recovered);
   (b) net control capture (control can't be recovered);
   (c) illegal distribution of control commands;
   (d) nodes blocking;
   (e) net removal.

Due to the defense system specific, some items cannot be considered as threats for them:

- 1.b: detection of the source of new nodes creation. In defense system new nodes usually aren't created automatically, so even if the source of new nodes creation at the particular moment is detected, it won't affect later nodes creation process;
- 1.e: nodes detection: botnet can treat all network nodes as nodes of distributed defense system, and in case, when all network is protected, this assumption will be correct.

Despite the fact, that defense systems also turn to distributed topologies, their topologies are more weak, comparing to botnets': star or star of stars (Kaspersky Security Network, ESET Live Grid, Panda Cloud Antivirus, etc.).

There are a lot of cases in which it is necessary to know, who will win in the "war" between NCDA and NCDA:

- when building the defense system;
- when comparing particular defense system and botnet;
- etc.

To be able to solve these problems, computer security experts need the model of interaction between NCDA and NCMA, which takes as an input description of two nets and as output gives the decision about winner.

2 Related Works

NCxAs are typically modeled by one of four types of models: imitation models, graph models, game models and stochastic models. Graph models represent only static states of modeled system. Imitation models give dynamic state representation, but don't allow to get analytical estimation of state parameters. Game models are rarely used because of difficulties with setting payoff function. Stochastic models allow both to represent dynamic system state and get both analytical and statistical estimations.

There are several approaches that measure botnet and defense systems parameters separately: in [1] it is proposed to measure botnet effectiveness due to the use of botnet, but these metrics couldn't be adopted for the general defense system. These approaches give detailed description for nets of cooperative agents, but don't consider the controversy between antagonistic groups of agents.

There are also approaches that rectify described flaws and study interaction between cooperative and antagonistic teams of agents [2]. In these models estimations are based on simulation mechanisms and don't give decision about the team win for class of nets, but just for one specifically defined example. So, existing approaches allow only to make an estimation about already existing net systems, when detailed description of its mechanism is given, and cannot be used to verify an architecture or structure of the new defense system, meant to be built.

Furthermore, there are approaches that develop a differential game model between the botnet herder and the defender group for simultaneous moves. Both players are strategic in their behavior, that is, they take actions that optimize their objective while also considering the actions of their opponent [3]. These approaches utilize game theory to model interactions between an attacker and a defender. But in this proposed model the profit of attacker is treated only in commercial way, which is not suitable for all botnets, because it is not always possible to estimate commercial value of information, collected by botnet. There are a couple of other papers, that also examine botnets from commercial point of view: [4] models botnet related cybercrimes as a result of profit-maximizing decision making from the economic prospectives of both botnet masters and renters/attackers, [5] discusses the economics of botnet in detail. Moreover, existing approaches to defense against botnets don't examine, whether defense system itself could be knocked out by botnet.

3 NxCA Model

The model of interaction between NCDA and NCMA takes as input description of the rival nets. NCxA state is represented by $\sum = (G, \Gamma)$. G is the graph of agents G(A, E), where A={$Agent_i$} is the set of agents (graph vertices) and E is the set of graph edges (connections between agents) or G(n, p), where n is the number of agents (graph nodes) and p is probability of connection existence between any two nodes.

New agents appear with the growth of NCxA and create connections with existing nodes. It represents installation of the defense system agent on other computers or spreading bots over the network. Also, at any time any agent can "die" and appropriate node and edge will be removed. It represents that fact, that agent was "killed" by the rival agent or some other actions on LAN environment: computer shuts off, network connection is lost, etc.

Γ is composition of evolutionary operators: $\Gamma=\Gamma_1\otimes\Gamma_2\otimes\Gamma_3\otimes\Gamma_4\otimes\Gamma_5$. Γ_1 represents creation of the new node with probability p_v, Γ_2 represents deletion of the node with probability q_v, Γ_3 represents creation of the new edge with probability p_e, Γ_4 represents deletion of the edge with probability q_e, Γ_5 represents node state transition.

Not all nodes in NCxA are equal: some possess control functionality. There is subnet Master $\subseteq$ A, which includes agents, which can control other agents, i.e. it can send message to others, containing malicious or defense task to execute, depending on its aim. Master nodes are also responsible for processing results of task execution.

Having the model for describing NCxA, it is possible to declare the model of interaction between NCDA and NCMA.

4 Stochastic Model of Interaction between NDCA and NMCA

In the opposition between two or more rivals there are two aspects: the power of weapon (in our case: the power of mechanisms, used for breaking confidentiality, availability and integrity) and the power of defense (in our case: the power of confidentiality, availability and integrity protection).

Nowadays for assessment of botnets and defense systems dissimilar characteristics are used. Defense systems are traditionally evaluated by false positive and false negative rates, networked distributed architecture of the system isn't taken into consideration [6-7]. For botnets their architecture is taken into consideration, but the set of characteristics is not full and cannot be adjusted for defense systems. Moreover, different characteristics are used to describe different usage types of botnets:

- giant portion. Large number of nodes increases the likelihood of high-bandwidth agents. Diurnal behaviour favors giant portion over total population;
- diameter. Agents sending messages to each other and coordinating activities require efficient communications;
- local transitivity. Agents maintaining state require redundancy to guard against random loss. Highly transitive networks are most robust [8].

Firstly, NCxAs were estimated by the size of its population (number of agents). But this characteristic doesn't take into consideration connectivity of the net: there could be a lot of alive agents, but without connections between them, NCxA won't work properly. After that, different graph characteristics were used to describe NCxAs: giant portion size, graph diameter, local transitivity. But these characteristics don't take into consideration such parameters, which are also essential for NCxA to function properly, as:

- deterministic time of message delivery;
- connectivity to control center;
- estimation of overhead expenses.

According to these facts, new characteristics were proposed, forming the model of interaction between NCDA and NCMA. Proposed model of interaction between NCDA and NCMA takes into consideration the power of defense and is based on four main factors of NCxA sustainability:

1. Controllability – $C(t)$ – the ability of NCxA to control its' nodes. Formally this ability is probability of the fact, that message, sent in NCxA from one node to another, will reach the destination in deterministic time *t*. Or, in other words,

probability of the fact, that between any nodes there is path of the length l_t, which can be passed in time t:

$$C(t) = (\sum_{i=0}^{n} \prod_{m=0}^{Mi} (1 - p_{im,t}))/\mathrm{n}\,, \tag{1}$$

where Mi is number of control centers, which can serve i-th agent, $(1 - p_{im,t})$ is probability of the fact, that the path between i-th node and control center exists, and message along this path can be transferred within time t. This probability depends on probabilities p_v, q_v, p_e, q_e, and can be calculated either analytically, if there is formal model, which describe this dependence, or by Monte Carlo methods.

Controllability is the main parameter, which describes the net of agents in non-hostile stable environment. But NCxA is believed to function in antagonistic environment, which aims at deleting nodes of this NCxA and this impact cannot be prevented with probability equals to 1. So, sustainable NCxA should stay controllable even if part of its' nodes is deleted and take measures to restore deleted or erased nodes. This property we will declare as resiliency.

2. Resiliency – R_{max} – is level of system instability, when is reached, takes system controllability to 0:

$$C_t(R_{max}) = 0 \tag{2}$$

where $R = (q_v, q_e)$, C_t is controllability for fixed time t, $0 < R < R_{max}$.

Previous two parameters directly represent NCxA behaviour under conditions of external purposeful influence. If NCxA implements any methods for protecting itself from such attacks, following parameters should also be taken into consideration:

3. Operational constancy – acceleration of traffic amount change,:

$$OC = \frac{\partial V}{\partial \mathrm{R}} \tag{3}$$

where $V(R)$ is the amount of traffic and R is level of instability. This metric reflects the NCxA impact on the normal operation of the local area network.

In addition to this we should consider that NCMA, as well as NCDA, is acting under conditions of adding and deleting nodes, so it should be scalable enough and its parameters, described above, will not significantly change after significant increasing or decreasing number of nodes in the net.

4. Scalability – dispersion of controllability, resiliency and operational constancy for NCxA of k to n nodes:

$$\mathrm{S} = (\sigma \mathrm{C}_{(\mathrm{k,n})}(\mathrm{t}), \sigma \mathrm{R}_{(\mathrm{k,n})}, \sigma \mathrm{V}_{(\mathrm{k,n})}(\mathrm{n}_{\mathrm{del}})) \tag{4}$$

This parameter describes the ability of NxCA to fulfill its functions correctly both for small and big nets.

Also it is assumed that interaction between agents in NCxA is cryptographically strong, that is MitM attacks are impossible. All listed characteristics should be treated in combination. For instance, net with centralized architecture, which is often implemented in defense systems, has the best rate of controllability equal to 1 – all nodes

are controlled by one central node. But its resiliency is R_{max} = 1/n, where n is number of nodes in the net, because deleting central node leads to losing controllability. So, in proposed model NCxA interaction is described with a vector:

$$IM = ((C(t), R_{max}, OC, S)_0, \dots, (C(t), R_{max}, OC, S)_m) \quad (5)$$

where *m* is the number of opposing nets.

5 Modeling Interaction between NxCA with Centralized Topology and NxCA with Random Topology

NCxA with centralized or hybrid architecture with small number of control nodes, which is often used by defense systems, has good controllability level, but is extremely non-resilient: if its' control centers are taken down, all net become unsustainable. In works, related to botnet architecture analysis, random architecture is said to be most sustainable to external attacks. Similar architecture can be applied to defense systems. To represent the interaction between NCxA with centralized topology and random topology, the proposed model of interaction will be used. It is assumed, that first NCxA has topology of classical random graph, so it can be described with Erdos-Renyi model. Reliability of Erdos-Renyi graph is described by the following theorem [9]:

Theorem 1: Consider there is model G(n, p). Let *p* be $p = \frac{c*\ln(n)}{n}$, where n is number of nodes in the net, *p* is probability that any two nodes have edge between them and *c* is constant. If $c > 1$, then assumption that random graph is almost always connected is true. If $c < 1$, then assumption that random graph is almost always non-connected is true.

Term "almost always" means that probability of some event seeks to 1, when $n \to \infty$.

Theorem 1 states that probability of saving graph connectivity, when edge removal probability is $q = 1 - p$, seeks to 1. That is, if there is net of 1000 nodes, then nodes can be deleted with probability close to 0,993, so that as a result, close to 1, interaction between any two nodes will be possible.

This theorem implies the following statement:

Theorem 2: Consider there is model G(n, p). NxCA on the random graph G is almost always controllable, if $p = \frac{c*\ln(n)}{n}$, $c > 1$.

The proof of Theorem 2 follows from declaration of controllability through connectivity: $C(t) = (\sum_{i=0}^{n} \prod_{m=0}^{Mi}(1 - p_{im,t}))/n$. Therefore, if the net is almost always connective (this follows from Theorem 1), it is also almost always controllable.

Also from Theorem 1 follows Theorem 3 [10]:

Theorem 3: Consider there is model G(n, p). Let $p = \frac{c*ln(n)}{n}$. Then, if $c > 3$, then for $n > 100$ following assumption is correct:

$$P_{n,p}(G \text{ is connected}) \geq 1 - \frac{1}{n} \quad (6)$$

This means, that if there is NxCA with the net of 1000 and probability of edge removal is $q = 1 - p = 1 - \frac{3*\ln(1000)}{1000} \approx 0{,}98$, probability of saving graph connectivity is not less than 0,999.

Theorem 1 is also interesting because there is a harsh jump from "almost always connectivity" to "almost always non-connectivity". Function $p = \frac{\ln(n)}{n}$ acts as a frontier, overcoming which means transition from connectivity to non-connectivity. Number of all edges can be calculated as $C_n^2 = O(n^2)$, probability of edge removal is $1-c(\ln(n)/n)$. Then, the expected number of not removed edges is about $n * \ln(n)$ and this number is enough for maintaining graph connectivity.

So for random graph percentage of the edges, that can be removed, is $R_{max} = (C_n^2 - n * \ln(n))/C_n^2$. That follows from probability of the edge removal $q = 1 - p$.

NCxA with centralized architecture has controllability C(t) = 1 – all nodes are controlled by one central node and resiliency $R_{max} = 1/n$. So, according to proposed interaction model, NCxA with random architecture is close to centralized by its controllability: $C(t) = (\sum_{i,j=1}^{n} \left(\frac{c*\ln(n)}{n}\right)^{l_t(i,j)} - n)/n$ and has much better resiliency: $R_{max} = (C_n^2 - n * \ln(n))/C_n^2$ against $R_{max} = 1/(n-1)$ for nets with centralized architecture.

6 Conclusion

Botnets are now evolving and there is actual problem of developing effective defense systems, which will be able to withstand targeted attacks, organized by botnets. But nowadays there is no model, allowing to predict the result of opposition between botnet and defense system. The proposed model of NCxA solves this problem and allow to compare different botnets and defense systems with each other. With the help of proposed NCxA model it is possible to build defense system model, capable of protecting itself from botnet attacks.

References

1. Dagon, D., Gu, G., Zou, C., Grizzard, J., Dwivedi, S., Lee, W., Lipton, R.: A Taxonomy of Botnet Structures. Botnet Detection 36 (2008)
2. Kotenko, I., Konovalov, A., Shorov, A.: Agent-based Modeling and Simulation of Botnets and Botnet Defense. Conference on Cyber Conflict, pp. 21–24. CCD COE Publications, Tallinn, Estonia (2010)
3. Bensoussan, A., Kantarcioglu, M., Hoe, S: A Game-Theoretical Approach for Finding Optimal Strategies in a Botnet Defense Model. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec 2010. LNCS, vol. 6442, pp. 135–148. Springer, Heidelberg (2010)
4. Li, Z., Liao, Q., Striegel, A.: Botnet Economics: Uncertainty Matters. In: The 7th Workshop on the Economics of Information Security (2008)
5. Namestnikov, Y.: The Economics of Botnets, http://www.viruslist.com/en/downloads/pdf/ ynam botnets 0907en.pdf

6. Bellaıche, M., Gregoire, J.C.: Measuring Defense Systems Against Flooding Attacks. In: International Wireless Communications and Mobile Computing Conference IWCMC 2008, pp. 600–605 (2008)
7. Wu, Z., Dong, H., Liang, Y., McKay, R.I.: A Chromosome-based Evaluation Model for Computer Defense Immune Systems. In: Proceedings of the IEEE Congress on Evolutionary Computation, Canberra, Australia, pp. 1363–1369 (2003)
8. Dagon, D., Gu, G., Zou, C., Grizzard, J., Dwivedi, S., Lee, W., Lipton, R.: A Taxonomy of Botnets (2010)
9. Erdos, P., Renyi, A.: On random graphs. Publ. Math. Debrecen (1959)
10. Raygorodskyi, A.M.: Random Graph Models. In: MIPT Proceedings, T. 2, №4 (8), pp. 130-140 (2010)

Malware Characterization Using Behavioral Components

Chaitanya Yavvari, Arnur Tokhtabayev,
Huzefa Rangwala, and Angelos Stavrou

Computer Science Department, George Mason University, Fairfax, VA, USA
{cyavvari,atokhtab,astavrou}@gmu.edu, rangwala@cs.gmu.edu

Abstract. Over the past years, we have experienced an increase in the quantity and complexity of malware binaries. This change has been fueled by the introduction of malware generation tools and reuse of different malcode modules. Recent malware appears to be highly modular and less functionally typified. A side-effect of this "composition" of components across different malware types, a growing number of new malware samples cannot be explicitly assigned to traditional classes defined by Anti-Virus (AV) vendors. Indeed, by nature, clustering techniques capture dominant behavior that could be a manifestation of only one of the malware component failing to reveal malware similarities that depend on other, less dominant components and other evolutionary traits.

In this paper, we introduce a novel malware behavioral commonality analysis scheme that takes into consideration component-wise grouping, called behavioral mapping. Our effort attempts to shed light to malware behavioral relationships and go beyond simply clustering the malware into a family. To this end, we implemented a method for identifying *soft* clusters and reveal shared malware components and traits. Using our method, we demonstrate that a malware sample can belong to several groups (clusters), implying sharing of its respective components with other samples from the groups. We performed experiments with a large corpus of real-world malware data-sets and identified that we can successfully highlight malware component relationships across the existing AV malware families and variants.

Keywords: Behavioral clustering, malware component analysis.

1 Introduction

The recent discoveries of sophisticated malware including Stuxnet [4] and Flame [9] demonstrate the evolution of the mainstream malware techniques to stealthy, precise cyber weapons aimed to disrupt critical infrastructure and exfiltrate sensitive information. To avoid mainstream Anti-Viruses (AVs) and intrusion detection systems, adversaries employ code obfuscation including polymorphism and metamorphism techniques to enshroud their malware attacks. At the same time, a new family of malware generator tools have gained popularity by offering

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 226–239, 2012.

capabilities for customization. The use of malware "components" has become evident with the recent leaks of malware source code (e.g. SpyEye [6] and Zeus [16]) that pointed to a modular structure of the malware development process, incorporating independent components into a new malware build. This observation indicates two implications: (i) modern malware should be viewed as a set of functional components; (ii) the number and diversity of functionally distinct components used in modern malware is rather limited.

Currently, the AV industry categorizes malware based on main malicious activities such as virus, worm, spyware, fakeAV and adware. As a result, malware samples were labeled and grouped based on one component, e.g., the one with the most threatening behavior (Kaspersky AV) [10]. In an attempt to account for expanding malware behavioral variety, several major AV companies adopted more detailed, tree-based malware classification [11]. Unfortunately, even this analysis is not adequate for labeling modern malware having multiple components attributed to various fixed types defined by the classification tree. For instance, bot frameworks offer a wide range of malicious functionality from self-replication to keylogger and backdoor, which traditionally belong to different malware types. Hence, MAEC project, the recent initiative of universal malware classification, proposes labeling and grouping malware based on the set of individual behaviors (components) to avoid class members inconsistencies [12].

On the other hand, researchers have proposed several methods that leverage various machine learning and clustering algorithms to group malware [8,2,15,1]. Although accurate when they come to single family, these methods fall short for modular malware: they obtain hard (exact) clusters that imply that each sample is attributed to one cluster (group). By nature, these clustering methods process totality of samples behavior and capture only dominant behavior that could be a manifestation of only one component. However, in practice, malware samples may share components with relatively small behavioral trace (footprint). Typical clustering will not reveal smaller, but potentially equally important, shared malicious components. Particularly, it may come short of exposing relationship between older malware and newer mixed samples.

In this paper, we propose to address the challenge of grouping malware with respect to components. In order to achieve component-oriented grouping, we developed a novel approach for building *soft* clusters that expose behavioral commonalities characterized as component traits. In our approach, a malware sample is decomposed to identified behavioral components and thus it can belong to several groups (clusters). Figure 1 illustrates the advantage of soft clustering for component-based malware grouping conceptually. The figure depicts three samples, each having two components. It could be seen that hard clustering cannot properly group samples with respect to components - sample 1 and 3 share components with sample 2 but not with each other. This component-wise orthogonality of samples 1 and 3 would render hard clusters to singletons (e.g. the sample 1 and 2 are clustered together, but sample 3 is excluded). In contrast, soft clustering allows for grouping samples appropriately as shown by dashed regions based on their behavioral similarities.

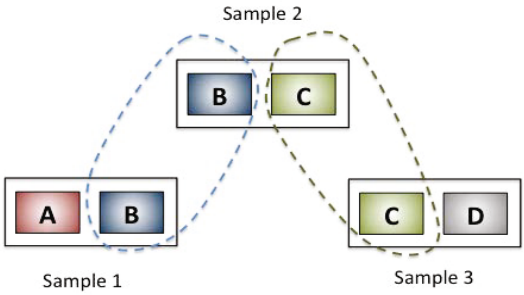

Fig. 1. Typical versus Component clustering. Typical clustering utilizes behavioral commonalities across the totality of the malware sample creating "hard clusters" that fail to capture smaller traits and behavioral sharing. On the other hand, "soft clusters" are designed to reveal all behavioral similarities, however small.

At a high level, the main concept behind our approach is called *behavioral mapping*, a process of rapid analysis of the commonalities between malware behavioral traces across large malware data sets. The behavioral map of a malware sample is produced by projecting its observed runtime behavior to the runtime behavior of another reference malware sample. The produced map is in essence a feature space defined by the behavioral projections and serves as a visualization mechanism for commonality sharing across analyzed samples. In our analysis, we generate malware behavioral maps and use a set of feature spaces to form soft clusters representing behavioral commonalities among samples.

In summary, this paper makes the following contributions:

1. **Component-based malware grouping.** We developed a novel approach for component-oriented, behavior-based malware clustering. We leverage what we call "soft clustering" to capture complex malware relationships with respect to all observed behavioral commonalities.
2. **Commonality analysis and visualization.** We introduced a behavioral mapping technique that allows for fast commonality identification, analysis, and visualization. Also such a map forms a feature space for sample clustering.
3. **Evaluation and real malware relationship interpretation.** We evaluated our system on substantial set of real-world malware - 1,727 unique samples. The experiments demonstrated that existing approaches for malware classification based on dominant functionality, i.e., AV labels [11], does not reveal real relationship between malware with respect to shared activity. Using our approach of "soft clusters", we were able to reveal malware relationships beyond basic family classification.

2 Malware Soft Clustering Using Behavioral Mapping

2.1 Behavioral Mapping

Behavioral mapping is the process of analyzing the commonalities between malware behavioral traces. Behavioral traces are sequences of system events collected from malware runtime observations. These traces are analyzed with the goal of identifying the commonalities which are subsequences that are shared among samples. These commonalities can lead to the exposure of shared components.

A behavioral mapping is produced in 3 steps: **(i) Projection:** Sequences of malware traces are projected onto a reference in the given domain. The reference can be another sequence of a sample or a constructed sequence of interest to the analyst. Projection is composed as a binary feature vector of length equal to reference length. It represents common behavioral sequences between a reference sample and the projected sample. **(ii) Soft Clustering:** Samples are clustered in the feature space defined by the projections on the reference. This provides an ordering of the samples as well as grouping of samples according to the similarity of their projections. **(iii) Visualization:** The behavioral map is presented as a bitmap of projections, viewed as commonalties, represented by rows and arranged according to clusters. Each row shows the shared behavior(shaded rectangles) of one malware sample with respect to the reference. We also visualize how much of the sample's observed behavior(sample coverage) is shared with the reference using an additional column on the map.

It is worth noting that this mapping approach can be used for analysis of any kind of behavioral sequence data irrespective of the abstraction level. In this work we chose to use only windows system events that could be monitored via Event Tracing for Windows(ETW) facility. Common behavioral sequences are identified by suffix tree based methods [18,5]. Inspection of behavioral maps can reveal interesting properties about the shared behavior of samples and their similarities.

2.2 Commonality Analysis via Iterative Behavioral Mapping

During the analysis of a corpus of malware samples, a single behavioral mapping will not suffice to elicit all the components present in the sample set. In a single mapping with one reference, all the samples similar to the reference or those that share a significant commonality with the reference show up together in groups. But, the rest of the samples which have low *coverage* remain mostly unexplored. They may have components they share between themselves but not with the reference. Therefore, these samples should be projected on to a new reference.

To address unexplored samples and identify all commonalities, we developed an iterative behavior mapping scheme that leverages soft clustering approach for commonality analysis and identification. To this end, samples are assigned to clusters in a fuzzy fashion (i.e multiple cluster assignments for the commonalities identified across multiple iterations). Such clusters represent commonalities that can enable component identification, e.g. via additional semantic analysis. To

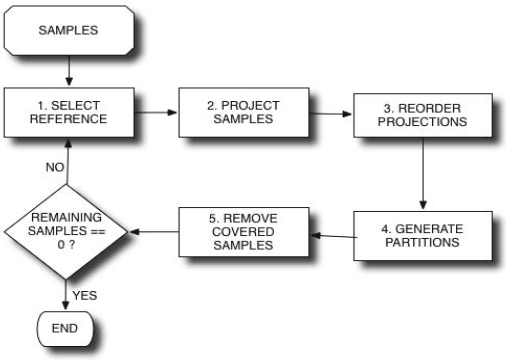

Fig. 2. Flow chart of Iterative Behavioral Mapping

quantify the progress of commonality discovery in samples, we use the following metrics for a given sample: (i) *sample coverage* - portion (%) of sample behavior shared with the reference as defined by the projection; (ii) *reference coverage* - portion (%) of reference behavior shared with the sample as defined by the projection. Figure 2 shows the flowchart of a procedure implementing our approach. The formal representation of the procedure is given in Algorithm 1. In each iteration, one of the samples from the pool of remaining samples is randomly chosen as a next reference. Next, we generate projections for each sample with respect to the selected reference. The generated projections are then ordered according to the result of Hierarchical clustering. The order of samples is the same as the leaves of the tree generated by hierarchical clustering. For ordering the projections(step 3), we first generate a pairwise distance matrix for all the projections. For this, we use a metric called $shared\,string\,metric(SSM)$ defined between every pair of sample projections as follows:

$$SSM[A,B] = 1 - 2 * ANDSimilarity(A,B)/(L(A) + L(B)) \quad (1)$$

where $L(A)$, $L(B)$ are sum of lengths of all shared strings of A, B respectively with reference.
$ANDSimilarity(A,B)$ = sum of lengths of all strings jointly shared by A and B with reference.

This metric captures the similarity of two projections with respect to a given reference. It is computed by the AND operation on the corresponding projection vectors both of which have the same length of the reference. Next, Hierarchical Agglomerative Clustering (HAC) is used as a method of clustering the SSM distance matrix. We used agglomerative nesting algorithm with wards linkage method. Agglomerative algorithms start with each of n samples as a separate cluster and iteratively merge the two nearest clusters in $n-1$ steps to produce a single hierarchical clustering. Wards linkage method is preferred over other linkage methods because it achieved higher *cophenetic correlation* with the input distance matrix in our experiments.

Algorithm 1. Iterative Behavioral Mapping.

Inputs: S : samples
T_s : sample coverage threshold
Definitions: P : pool of remaining samples
R : List of references
Cl_{ji} : j th cluster in i th iteration
Pr_i : set of projections of samples in the i th iteration
Co_i: set of coverages of samples in the i th iteration
Co_{ki}: coverage of sample k in the i th iteration
Cc_i: set of cumulative coverages of samples in all iteration until ith
Cc_{ki}: cumulative coverage of sample k after i iterations
r_i : reference in the i th iteration
s_i : i th malware sample
p_{ki} : projection of the k th sample in the i th iteration
Initialization :
$P = S$ ▷ initial pool is the whole set of samples
$i = 1$

1: **procedure** ITERATIVE PROJECTION(S, P, T_s) ▷ To perform iterative projection
2:
3: **while** ($|P| \geq 2$) **do** ▷ sample pool size is at least 2
4: $r_i \leftarrow$PICK REFERENCE(R, P, i)
5: $P \leftarrow P - r_i$ ▷ remove reference from pool
6: $\forall s_k \in P : pr_{ki} \leftarrow$Project Sample($s_k$,$r_i$) ▷ projection of the k th sample on the reference
7: $Pr_i = \{pr_{ki} \mid \forall k = 1\ to\ |P|\ \}$ ▷ all projections
8:
9: * *cluster all projections* *\
10: ClusterHAC(Pr_i) ▷ generates hierarchical clustering of projections
11: **for all** $s_k \in P$ **do**
12: UPDATE COVERAGE(Cc_{ki},Co_{ki},pr_{ki}) ▷ update sample coverage
13: **if** $Cc_{ki} \geq T_s$ **then**
14: $P \leftarrow P - s_k$
15: **end if**
16: **end for**
17: **end while**
18: $i \leftarrow i + 1$
19: **end procedure**

In step 4, the dendrogram generated in step 3 is partitioned into separate clusters. These clusters of projections essentially represent *soft* clusters of corresponding samples. To partition the projections into separate clusters (step 4), we use the *Dynamic hybrid tree cut* method [13] with minimum cluster size set to 1. The dynamic hybrid tree cut method performs better than fixed height cutting for partitioning a hierarchical clustering result into separate clusters because it incorporates the tree structure information into the partitioning method. After partitioning, samples within each cluster have comparable *reference coverage* as seen on the reordered behavioral maps. But, consistent *sample coverage* is not guaranteed (i.e projections look similar but samples themselves need not be similar). In the last step, we assess each sample's coverage and remove highly covered samples from further iterations. Based on the Threshold T_s set in Algorithm 1, we eliminate samples that have accumulated coverage above the threshold. This step reduces the pool size for the next iteration.

3 Evaluation

We evaluated our approach on a real-world data set containing 1,727 samples from seven families. Table 1 shows the distribution of samples in our sample set according to their *Kaspersky AV* family labels. In section 3.1 we present two use cases of behavioral mapping for analysing variants of a single family and multiple families respectively. In Section 3.2 we present the evaluation of the Iterative behavioral mapping scheme described in Section 2.2.

Table 1. Malware binaries distribution by Kaspersky Family

Distribution by Kaspersky Family	
BackdoorSdBot	25
BackdoorSdBot-05	5
BackdoorXtoober	329
Trojan-SpyZbot	411
TrojanBuzus	392
TrojanMenti	205
TrojanRefroso	629

3.1 Behavioral Map Use Cases

Intra-family Mapping : In this experiment, we use behavioral mapping to illustrate the properties of the variants of a given family. We call this *intra-family* map because all samples belong to the same family. Figure 3 shows the Behavioral Map for 92 samples of the *Trojan Jorik* Family (Kaspersky AV definition). We randomly selected a sample which had label *Trojan.Win32.Jorik.SpyEyes.pq* as a reference. The column adjacent to the projection represents *Sample coverage*. The projections are clustered into 7 clusters according to their similarities.

From the labels (on Y-axis), we can see that the Behavioral map reveals clusters of samples that are consistent with Kaspersky AV labels(i.e samples belonging to same variant are clustered together). We also observe that length of shared behaviors is invariant across many samples. These are likely to be caused by shared components. It can be observed from the map that several components exclusively belong to certain variants. In Figure 3, we can see that component marked **C1** occurs in most samples except the *Fraud* variant and component marked **C2** is never present in the *Gbot* variant. From the column showing sample coverages, we can observe that the samples with highest coverage belong to the same variant as the reference (*SpyEyes*). From the low coverages of other samples we can infer that they exhibit behaviors not shared with *SpyEyes* variant.

Inter-family Mapping - Composite Reference : In this experiment, we produce a behavioral map for a set of 337 samples representative of seven families. To analyze sample relationship across families we use composite reference in mapping. The composite reference is constructed in a supervised manner by randomly selecting a sample from each of the 6 families presented in the mapped sample set. Figure 4 shows the produced behavioral map.

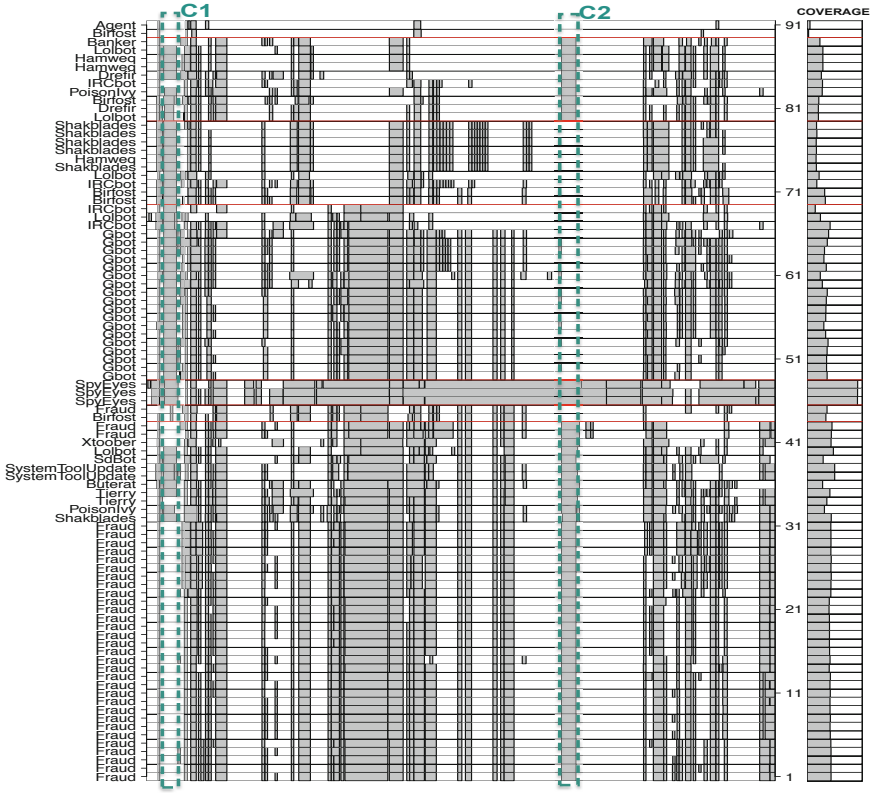

Fig. 3. Behavioral mapping of *Jorik* family variants

As indicated in Figure 4, some samples are highly covered by commonalties. High *sample coverage* and high commonality density indicates that samples are highly similar to the reference sample (e.g. cluster 4). High *sample coverage*, but commonalities themselves are short and scattered across the projections may indicate that these samples did not execute real malicious components exposing only typical (normal) system activity (e.g. dll loading), which we treat as noise from clustering perspective (e.g. cluster 11, 12). Low *sample coverage* indicates that these samples do not share much of behavior with the reference and should be projected onto another sample (e.g. cluster 6).

As depicted in Figure 4, the map provides significant visual and structural information for inter-sample commonality analysis. While the gain from each mapping of a sample set is subjective on the selected reference point, producing multiple mappings with various references or using the composite reference technique would increase its value for an expert from an analytical standpoint. In Figure 4, vertical lines separate the 6 individual behavior sequences. From this perspective, it could be seen that samples in cluster #7 share commonalties with various samples from various families. At the same time, samples in cluster

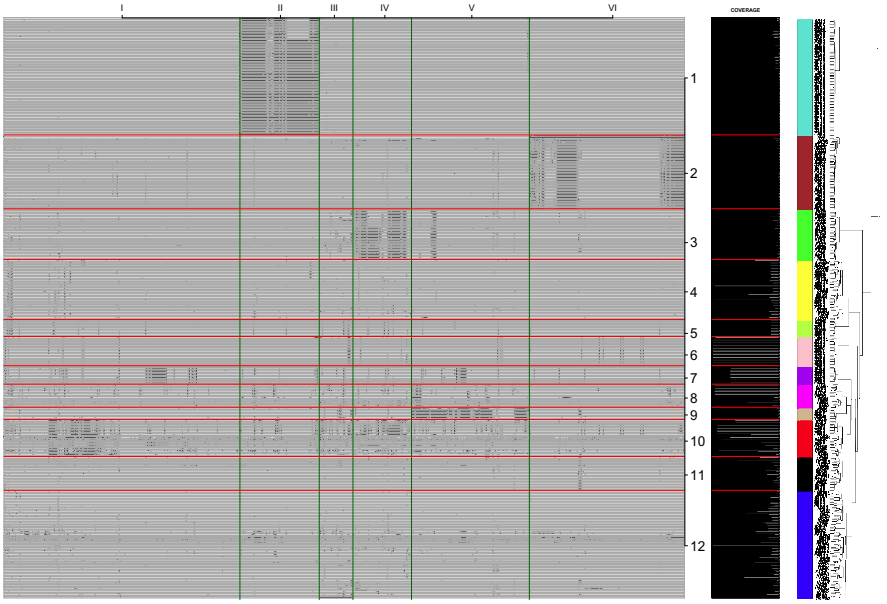

Fig. 4. Clustered commonality map with composite reference

#1 share almost entire observed behavior with the constituent sample reference II, meaning that they belong to a single highly consistent family.

3.2 Commonality Analysis (Iterative clustering)

In this section we perform malware commonality analysis for 1727 sample set using proposed Iterative Mapping method (Algorithm 1) [1]. In the experiment, we set coverage threshold $Ts = 90\%$, which means that the algorithm must process at least 90% of sample behavior. In spite of the high coverage threshold, the algorithm took only 38 iterations and identified 303 commonalities(soft clusters).

In Figure 5, we present the result of entire iteration process for a subset of samples (limited by visualization space). The **x** axis represents the concatenation of all commonalities found across iterations in increasing order. The **y** axis shows 337 samples arranged in the order of the number of iterations required to uncover sample behavior, i.e. achieve coverage threshold. The vertical partitions mark the end of iterations. It can be seen that more commonalities are revealed in the earlier iterations and the number of commonalities decreases in subsequent iterations. The commonalities also occur in larger groups in earlier iterations

[1] In this experiment, after the HAC based partitioning step in an iteration, we define a sample S to belong to its iteration-level cluster, which is indexed by the pair *(Iteration, clusterId)*, if the *sample coverage* is significant (e.g., more than 10%). In other words, the samples with insignificant coverage are not clustered in the current iteration.

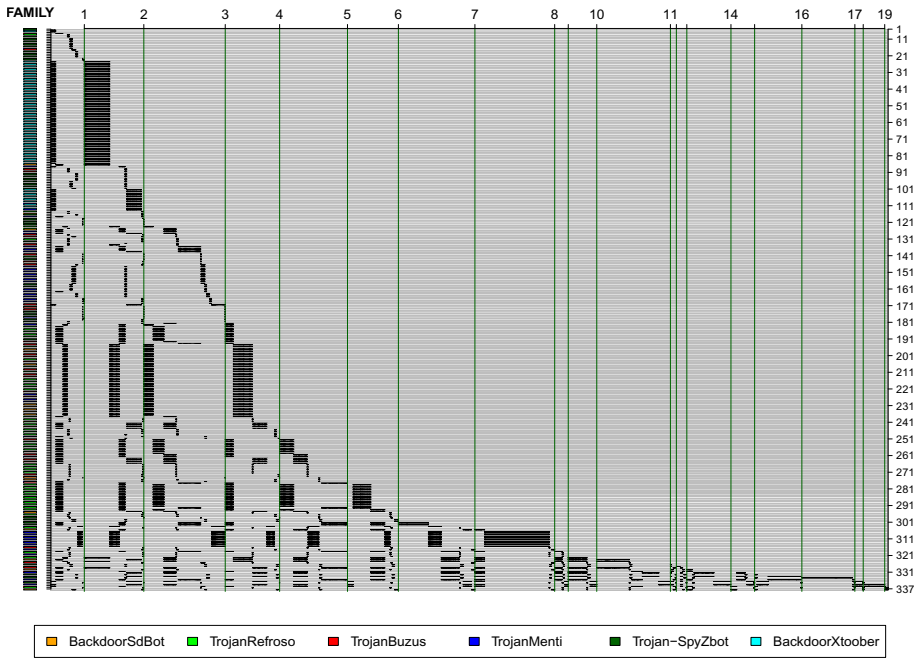

Fig. 5. Commonalities Across All Iterations

due to the higher number of samples in the pool. This figure essentially provides an approximate summary of the entire process across iterations. It also reveals groups of samples that share commonalities and are covered together.

Figure 6 presents a heat map depicting commonality sharing (cluster composition) among samples of various malware families for the entire set of 1,727 malware samples. The columns of the heat map represent the commonalities found across iterations. The rows divide each of the associated sample groups according to the family labels. The intensity of the gray scale color represents the purity of commonality sharing with respect to the *malware families*. If a cell is colored black it means the total absence of the component in the corresponding family. On the other hand, white indicates that the component is exclusive to the family. Intermediate shades of grey indicate the various proportions of families sharing the component. The same information is also presented by the horizontal trace across the rows.

It can be observed that some commonalities are exclusive for particular families, while others are shared across the families. This is indicative of the component sharing nature of different families. For example, it can be seen that samples from *Trojan.win32.Refroso* and *Trojan.win32.Buzus* families likely share some components (manifested as commonalities) . At the same time, samples from *Trojan.win32.Refroso* and *Trojan-Spy-win32.Zbot* families potentially share some other components.

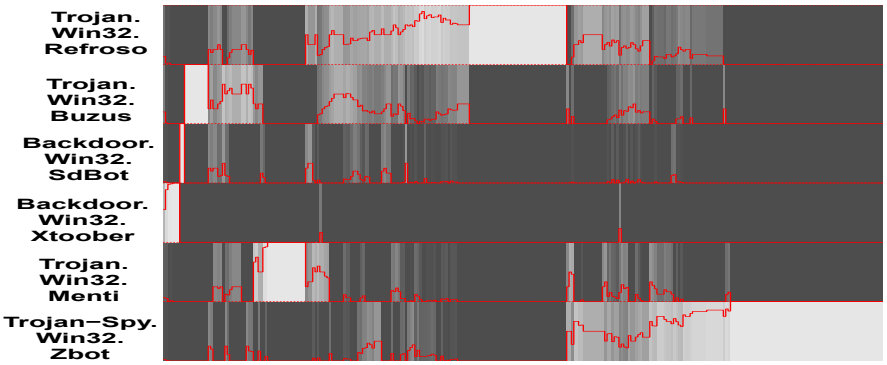

Fig. 6. Inter-Family Commonality sharing

In Figure 7, we present the graph-based visualization of the clustering results. It shows the structural relationship between samples from the commonality perspective. The graph shows two types of nodes: (i) samples and (ii) references of maps over all iterations. The sample nodes are colored according to their *Kaspersky Antivirus* label. The references are plotted in red. Semantically, the graph shows cluster membership and members proximity with respect to shared behavior (components). The samples that are grouped together are connected to the same reference representing shared commonality. The distance between references on the graph is proportional to SSM similarity described in 2.2 . The distance between samples and the corresponding reference is proportional to their coverage (*high sample coverage* means *low* distance).

For the sake of clarity, in Figure 7, we minimized the number of links by considering only the most significant components. It could be seen that some samples are all at the same distance to the reference and are homogeneously colored. These samples are similar to each other and belong to the same *Kaspersky* family. On the other hand other clusters have samples from different families, this also shows the inter-family behavior sharing property. Also, these clusters have samples at different distances from their respective references, this means that they share commonalities of different degrees with respect to each other.

To conclude our findings, we evaluated performance of hard clustering with behavioral maps. To build hard clusters we used only one map providing the sufficient coverage of samples, in contrast to several maps contributing total coverage as with soft clusters. To this end, samples that are not covered above minimum coverage in any iteration remain unassigned and continue to be in the sample pool for the next iteration. We observed there are many samples which were not covered beyond threshold in any single iteration and therefore not assigned to any hard cluster, however the same exact samples were almost completely covered by soft clustering approach and as a result assigned to multiple groups. This illustrates the problem with *one-one (pairwise)* comparison of samples for clustering. Though there is a component sharing, the sharing behavior is not captured by pairwise methods. This experiment shows that sample behavioral

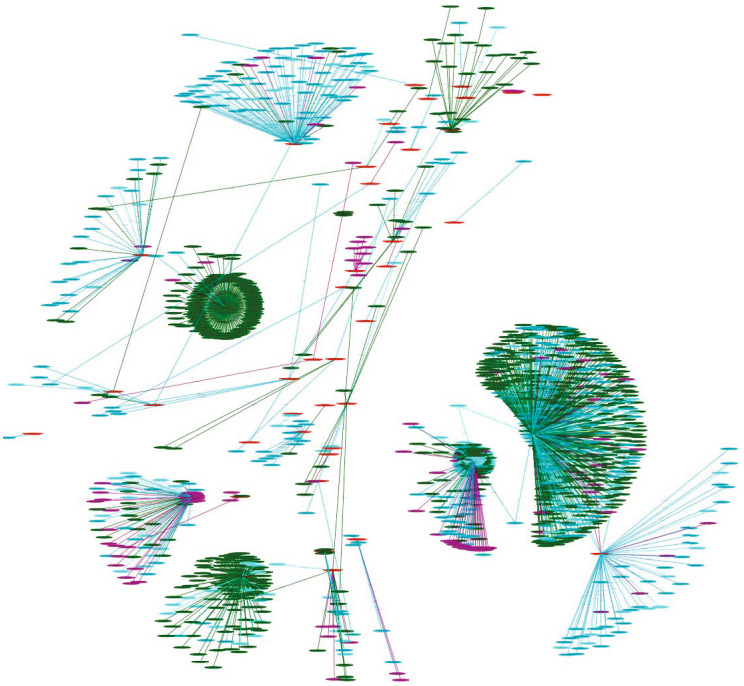

Fig. 7. Graph-based cluster visualization

sequences are indeed composed of distinct behavioral sub sequences (commonalities), that are shared with other samples. All of these commonalities cannot be extracted jointly in any single pairwise comparison. Finally, the iterative behavioral mapping scheme avoids $O(n^2)$ comparisons between all samples to extract these commonalities. It took only 38 mapping iterations to reveal commonalities and group all of the 1727 malware samples while the vast majority of the samples were analyzed and clustered during the first 10 iterations.

4 Related Work

There has been a great deal of research on development of dynamic malware analysis techniques. Egele et al. [3] provide a detailed survey of various existing dynamic malware analysis systems and a comparison of their analysis inputs and capabilities. Jacob et al. [7] present a taxonomy of behavioral detection methods according to the reasoning techniques deployed in them. While dynamic malware analysis allows for extracting samples behavior, our work is dedicated to processing the behavior.

Malware clustering in the behavioral domain was addressed in various publications [8], [2], [15], [1] and [14]. Most of the proposed approaches utilize standard clustering algorithms and focused on selection of appropriate feature space and

distance metric. By nature, such clustering approaches have a limitation called "dominance" effect, as the result the hard clusters may not reveal smaller but equally important malicious components.

Bayer et al. [2] , Rieck et al. [15] and Jang et al. [8] all work with behavioral profiles generated by processing execution reports and generating feature sets. These works focus on scalable clustering by incorporating suitable approximations. BitShred [8] performs feature hashing and co-clustering to reveal semantic relationships between families. Their method requires preselected feature extraction and operates on vector data. Also, due to co-clustering it could not be applied to ordered sequence data for semantic analysis. Our system is feature order sensitive and preserving, allowing for direct analysis of behavior data, such as operation/function call sequences of dynamic length.

Rieck et al. [15] and Trinius et al. [17] extract behavioral profiles of malware samples from CWSandbox reports. They generate feature vectors based on *n-grams* from these reports and perform clustering to find groupings(*class discovery*) and classification(using SVM) to assign unknown malware to known classes. Because our system is not *n-gram* feature space based, it is scalable for use with any sequence data irrespective of the alphabet size. Trinius et al. [17] visualize the CWSandbox reports in the form of treemapping and thread graphs, they perform visual malware clustering by generating tree maps for samples and evaluating against AV labels. Wagener et al. [19] and Bailey et al. [1] tackle the problem of automated classification of malware based on behavioral analysis using normalized compression distance(NCD) metrics. Ye. et al. [20] proposed an ensemble method to generate consensus of multiple clusterings using static features of unpacked malware.

5 Conclusions

Malware classification techniques are not new and there has been a lot of research into placing malware samples into different families including work by AV vendors. We focused on the problem of component oriented malware grouping. We used our "soft clustering" approach to reveal component sharing across malware samples that belong to different families according to traditional grouping. We experimentally demonstrated that existing approach for malware grouping based on dominant functionality and fixed classification tree, as used in AV industry, does not reveal relationships between malware with respect to shared behavior. We introduced behavioral mapping approach that iteratively builds a range of features which form soft clusters representing shared component traits. Furthermore, we used visualization for a set of samples to illustrate the structural commonality distribution across AV families. Finally, our experiments show the scalability and computational efficiency of our component analysis scheme on a real set of 1727 malware samples.

Acknowledgements. This work was partially supported by DARPA Cybergenome project through contract FA8750-10-C-169. The views expressed are

those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Bailey, M., Oberheide, J., Andersen, J., Mao, M., Jahanian, F., Nazario, J.: Automated Classification and Analysis of Internet Malware (2007)
2. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, Behavior-Based Malware Clustering. In: NDSS (2009)
3. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. 44(2), 6:1–6:42 (2008)
4. Falliere, N., Murchu, L.O., Chien, E.: W32.stuxnet dossier, White paper (2011), www.symantec.com
5. Gusfield, D.: Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology. Cambridge University Press (1997)
6. IOActive. Reversal and Analysis of Zeus and SpyEye Banking Trojans. Technical report, IOActive (2012)
7. Jacob, G., Debar, H., Filiol, E.: Behavioral detection of malware: from a survey towards an established taxonomy. Journal in Computer Virology 4, 251–266 (2008), doi:10.1007/s11416-008-0086-0
8. Jang, J., Brumley, D., Venkataraman, S.: Bitshred: feature hashing malware for scalable triage and semantic analysis. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 309–320. ACM (2011)
9. The flame: Questions and answers (May 2012), www.securelist.com
10. New malware classification system, www.securelist.com (accessed, June 2012)
11. Rules for naming detected objects, www.securelist.com (accessed, 2012)
12. Kirillov, I., Beck, D., Chase, P., Martin, R.: Malware attribute enumeration and characterization
13. Langfelder, P., Zhang, B., Horvath, S.: Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. Bioinformatics 24(5), 719–720 (2008)
14. Li, P., Liu, L., Gao, D., Reiter, M.K.: On Challenges in Evaluating Malware Clustering. In: Jha, S., Sommer, R., Kreibich, C. (eds.) RAID 2010. LNCS, vol. 6307, pp. 238–255. Springer, Heidelberg (2010)
15. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic analysis of malware behavior using machine learning. Journal of Computer Security 19(4), 639–668 (2011)
16. RSA. The Current State of Cybercrime and What to Expect in 2012. Technical report, RSA (2012)
17. Trinius, P., Holz, T., Gobel, J., Freiling, F.C.: Visual analysis of malware behavior using treemaps and thread graphs. In: 2009 6th International Workshop on Visualization for Cyber Security, 33–38 (2009)
18. Ukkonen, E.: Constructing suffix trees on-line in linear time. In: IFIP Congress (1), pp. 484–492 (1992)
19. Wagener, G., State, R., Dulaunoy, A.: Malware behaviour analysis. Journal in Computer Virology 4(4), 279–287 (2007)
20. Ye, Y., Li, T., Chen, Y., Jiang, Q.: Automatic malware categorization using cluster ensemble. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2010, pp. 95–104. ACM, New York (2010)

MADAM: A Multi-level Anomaly Detector for Android Malware*

Gianluca Dini[1], Fabio Martinelli[2], Andrea Saracino[1,2], and Daniele Sgandurra[2]

[1] Dipartimento di Ingegneria dell'Informazione
Università di Pisa, Pisa, Italy
firstname.lastname@iet.unipi.it
[2] Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche, Pisa, Italy
firstname.lastname@iit.cnr.it

Abstract. Currently, in the smartphone market, Android is the platform with the highest share. Due to this popularity and also to its open source nature, Android-based smartphones are now an ideal target for attackers. Since the number of malware designed for Android devices is increasing fast, Android users are looking for security solutions aimed at preventing malicious actions from damaging their smartphones.

In this paper, we describe *MADAM*, a Multi-level Anomaly Detector for Android Malware. MADAM concurrently monitors Android at the kernel-level and user-level to detect real malware infections using machine learning techniques to distinguish between standard behaviors and malicious ones. The first prototype of MADAM is able to detect several real malware found in the wild. The device usability is not affected by MADAM due to the low number of false positives generated after the learning phase.

Keywords: Intrusion detection, Android, Security, Classification.

1 Introduction

In the last years, mobile devices, such as smartphones, tablets and PDAs, have drastically changed by increasing the number and complexity of their capabilities. Current mobile devices offer a larger amount of services and applications than those offered by personal computers. At the same time, an increasing number of security threats targeting mobile devices has emerged. In fact, malicious users and hackers are taking advantage of both the limited capabilities of mobile devices and the lack of standard security mechanisms to design mobile-specific malware that access sensitive data, steal the user's phone credit, or deny access to some device functionalities. In 2011, malware attacks increased by 155% across all platforms [1]: in particular, Android is the platform with the highest malware growth rate by the end of 2011.

* The research leading to these results has received funding from the EU FP7 under grant n. 256980 (NESSoS) and under grant n- 257930 (Aniketos).

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 240–253, 2012.

To mitigate these security threats, various mobile-specific Intrusion Detection Systems (IDSes) have been recently proposed. Most of these IDSes are *behavior-based*, i.e. they do not rely on a database of malicious code patterns, as in the case of *signature-based* IDSes. A behavior-based (or anomaly-based) IDS is a system that attempts to learn the normal behavior of a device. To this end, the system is firstly trained by receiving as input a set of parameters that describes the way the user normally behaves. Secondly, during the normal usage, the IDS is able to recognize as suspicious any behavior that strongly differs from those well-known, i.e. learnt during the first phase.

In this paper, we describe *MADAM*, a Multi-level Anomaly Detector for Android Malware, which monitors Android both at the kernel-level and user-level to detect real malware infections. MADAM exploits machine learning techniques to distinguish between standard behaviors and malicious ones. A first prototype of MADAM has been implemented for Android smartphones, but its theoretical approach can be extended to other mobile operating systems (OS) as well. The first set of results show that this approach works well with real malware and it is usable since it has a very low false positive rate.

The main contributions of the paper are the following:

- We describe the design and implementation of MADAM, a host-based *real-time* anomaly detector that exploits a *multi-level view* of the monitored smartphone, which considers both OS events, namely the issued system calls, and smartphone parameters, e.g. the user activity/idleness, to detect intrusion attempts.
- We show that a dataset with a *small number of parameters* (13 features), and a relatively small number of elements, is effective in describing the smartphone behavior to a machine learning system; furthermore, MADAM can *self-adapt* to new behaviors by including new elements in the training set learnt at run-time.
- The framework has been implemented and tested on *real devices* (Samsung Galaxy Nexus) to understand the users' experience. The tests have been performed with more than 50 popular applications and several user behaviors to measure the false positives; on the average, a user receives less than 5 false positives per day, and the overall performance overhead is acceptable, i.e. 3% of memory consumption, 7% of CPU overhead and 5% of battery.
- To the best of our knowledge, MADAM is the first anomaly-based IDS for Android that has been tested using *real malware*: furthermore, at the time of the tests, some of the tested malware were *zero-day-attacks* and current off-the-shelf security solutions were not able to detect them. The system shows a detection rate of 93%, and in particular of 100% with rootkits.
- MADAM is able to detect *unwanted outgoing SMSes* stealthily sent by Android malicious applications.

The rest of the paper is organized as follows. Section 2 lists some related work. Section 3 describes the MADAM architecture and its current implementation. Section 4 reports some preliminary tests and results. In Sect. 5 we discuss the

features and the current limitations of the framework. Finally, Sect. 6 concludes by discussing some future works.

2 Related Work

Crowdroid [2] is a machine learning-based framework that recognizes Trojan-like malware on Android smartphones, by analyzing the number of times each system call has been issued by an application during the execution of an action that requires user interaction. A genuine application differs from its trojanized version, since it issues different types and a different number of system calls. Crowdroid builds a vector of m features (the Android system calls). Differently from this approach, MADAM uses a global-monitoring approach that is able to detect malware contained in unknown applications, i.e. not previously classified. Furthermore, on *Crowdroid* only two trojanised applications have been tested, whereas on MADAM we tested ten real malware. A similar approach is presented in [3], which also considers the system call parameters to discern between normal system calls and malicious ones.

Another IDS that relies on machine learning techniques is *Andromaly* [4], which monitors both the smartphone and user's behaviors by observing several parameters, spanning from sensors activities to CPU usage. 88 features are used to describe these behaviors; the features are then pre-processed by feature selection algorithms. The authors developed four malicious applications to evaluate the ability to detect anomalies. Compared to Andromaly, MADAM uses a smaller number of features (13), and has been tested on real malware found in the wild, and shows better performance in terms of detection and, especially, of false positives rate. After the learning phase, the false positive rate of MADAM is 0.0001, whereas that of [4], which uses a sampling method similar to that of MADAM and with a comparable sampling rate (2 seconds), is 0.12. The detection rate of MADAM is 93%, while that of [4] is 80%.

Other approaches only monitor misbehaviors on a limited number of functionalities such as outgoing/incoming traffic [5], SMS, Bluetooth and IM [6], or power consumption [7] and, therefore, their detection accuracy is higher of other work but less general. [8] monitors smartphones to extract features that can be used in a machine learning algorithm to detect anomalies. The framework includes a monitoring client, a *Remote Anomaly Detection System* (RADS) and a visualization component. RADS is a web service that receives, from the monitoring client, the monitored features and exploits this information, stored in a database, to implement a machine learning algorithm. In MADAM, the detection is performed locally and, more importantly, in real-time. [9] proposes a behavior-based malware detection system (*pBMDS*) that correlates user's inputs with system calls to detect anomalous activities related to SMS/MMS sending. MADAM is more general since it considers all the activities on a smartphone. A further framework targeted at SMS/MMS monitoring is *Proactive Group Behavior Containment* [10], which is aimed at containing malicious software spreading in these messaging networks.

[11] and [12] propose *Kirin* security service for Android, which performs lightweight certification of applications to mitigate malware at install time. Kirin certification uses security rules that match undesirable properties in security configuration bundled with applications. [13] performs static analysis on the executables to extract functions calls usage using `readelf` command. Hence, these calls are compared with malware executables for classification. Finally, [14] surveys some security solutions for mobile devices.

3 MADAM Approach

MADAM is a Multi-level Anomaly Detector for Android Malware that concurrently monitors Android at the kernel-level and user-level to detect real malware infections using machine learning techniques to distinguish between standard behaviors and malicious ones. In fact, the problem of anomaly detection can be seen as a problem of binary classification, in which each normal behavior is classified as "Standard", whereas abnormal ones are classified as "Suspicious". Some behavior-based IDSes rely on computational intelligence and machine learning techniques, such as clustering [2], probability-based classifiers [4] [5], decision trees [5] and others. Henceforth, we will use the generic term "classifier" for these techniques.

Classifiers automatically learn how to classify a set of items. In the proposed scenario they could be seen as a black-box whose input is a set of behaviors and the output for each behavior is "Standard" or "Suspicious". A classifier understands how to correctly classify elements after the execution of a training phase. This phase is critical, since it determines the accuracy of the classifier. Hence, it is fundamental to provide the classifier with a good training set.

To build a good dataset for smartphones, i.e. one that represents a typical smartphone behavior, MADAM considers elements that represents behaviors both when the user is active and when she is idle. Moreover, our training set also contains some malicious behaviors, which strongly differ from the standard ones. Usually, the collected features come from several sources of events [4]: choosing the right features to best represent the smartphone behaviors is a critical task, since their number and correlation determine the quality of the training set [15]. As discussed in Sect. 3.1, MADAM considers two levels, the kernel-level and the application-level.

3.1 Multi-level Detection

MADAM is a Multi-level Anomaly Detector for Android Malware that combines features extracted from several levels to (i) provide a wider range of monitored events and (ii) discover correlations among these events belonging to distinct levels. Currently, MADAM considers two levels, the kernel-level and the application-level. At the first level, MADAM monitors system calls. In fact, we believe that system calls are a good representative sample of the smartphone behavior, since their usage is a monitor for user activity, files and memory access, incoming/outgoing traffic, energy consumption and sensors status. More

importantly, they can be used as monitors for intrusion attempts: this is based upon the assumption that an attacker has to execute one (or several) system calls to harm the system. At the second level, the extracted features consider whether the user is idle or not, and the number of sent SMSes. A high-level view of MADAM architecture is depicted in Fig. 1.

To extract features from these two levels, the framework includes two monitors. The first one is a kernel-level monitor that intercepts all the critical system calls, and that records the number of their occurrences during a period T. Hence, if m is the number of monitored system calls, this monitor returns a vector of dimension m at each period T.

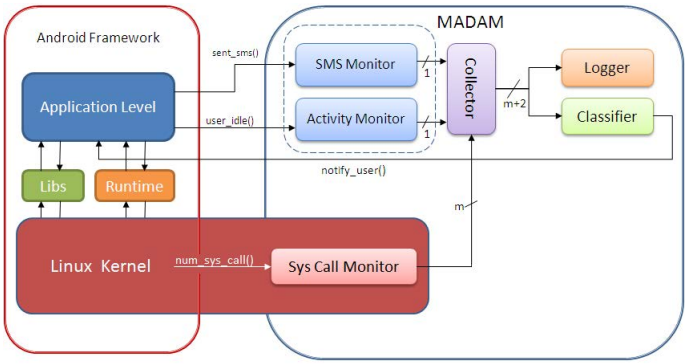

Fig. 1. Functional Blocks of MADAM

The second monitor is at the application-level, and it can be split in two sub-monitors that handle two different tasks: (i) to periodically measure the number of SMS sent in a time interval; (ii) to monitor the user idleness. The user idleness is a fundamental feature since the activity of the device is usually more intense when the user is interacting with the device itself: hence, the number of issued system calls depends upon the status of the device/user. Since after a very short period of user inactivity the smartphone screen is turned off by the OS, the user can be considered active either if the screen is on or a voice call is active [16].

The elements of the datasets are vectors with $m + 2$ features, where m is the number of monitored (critical) system calls and the last two features represent, respectively, the device status (idle or active) and the number of sent SMSes. A *collector* receives these features from all the monitors and then builds the vectors. These vectors are stored in local files using a *logger* module so that they can be used to build a training set, which is composed of $\frac{t}{T}$ vectors, where t is the total time spent collecting data and T is the logging interval (an input parameter of the framework). A training set is then used to obtain a trained *classifier*. This phase of data gathering, preprocessing and classifier training, is called the *Training Phase*. In the *Operative Phase*, which is the phase where the user actually uses the smartphone, each monitored vector is given as input to the

trained *classifier* and, if it is classified as *suspicious*, a notification is immediately shown to the user.

3.2 Implementation

We have developed the framework on a *Samsung Galaxy Nexus* HSPA, with OS Android *Ice Cream Sandwich* version `4.0.1`, and Linux kernel version `3.0.1`. The lowest-level component of MADAM framework is the system call monitor, which has been implemented as a Linux kernel module that hijacks the execution of the monitored system calls: each system call is coupled with a counter that is incremented before its execution. In the current implementation, this module considers only a subset of all the available system calls on Android Linux, those that are rather critical, in term of security, in the description of the system behavior (see Sect. 4.1). The kernel module contains a task that periodically (with a period of T) logs the actual value of the counters on a shared buffer with the collector and then resets all the counters. The inclusion and execution of the hijacking module requires the Super User (SU) permissions: since on the Android production builds (the OS version installed on device by manufacturers) SU is disabled, during the tests the devices required *rooting*, which is a procedure to get root permissions.

The highest-level component of the framework includes an Android Application in Java, which has been implemented using the Android SDK. The first component of the Java Application is the MADAM *collector*, which periodically reads (i) the buffer shared with the kernel monitor, (ii) the user status (idle/active), (iii) the number of SMSes sent in the period T. Since Android only allows monitoring SMSes that are sent through the default SMS manager, i.e. an application can send SMSes without the user being notified, to detect sent SMSes MADAM exploits the Android system log file (*LogCat*), which contains the output of a low level function that is called each time an SMS is being sent. Furthermore, the Java application also includes two parallel tasks. The first one is the application-level *logger* (Figure 1), which reads the vectors built by the collector and logs them in a log file that results in a matrix with $\frac{t}{T}$ rows. The second task is the *classifier* that states if the vectors built by the *collector* are good or suspicious. In the latter case, the classifier sends a notification to the user and logs those vectors that have been classified as suspicious, for further analysis. For classification we used Weka[1] version 3.6.6, an open source library in Java that includes several classification tools.

4 Experimental Results

In this section we describe in detail the tests which were performed both for malware detection and false positives measurement.

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

4.1 Training Set and Classifiers

To do so, we have logged the behavior (through system calls) of the phone during the execution of normal actions performed by a user. In this logging phase we have tried to ensure that the device has not been infected: we have installed only popular applications from the official site (Google Play) having a high rating and positive comments[2].

After a first set of preliminary tests, we have noticed that the system calls that best describe the device behavior are the following: `open`, `ioctl`, `brk`, `read`, `write`, `exit`, `close`, `sendto`, `sendmsg`, `recvfrom`, `recvmsg`. We expected such a result, since Android is a framework composed by several functional blocks that communicate using the mechanisms provided by the underlying Linux kernel and an increase in the smartphone activity causes directly a sharp increase in the occurrences of these system calls, all of which concern buffer or file operations, or communications between the framework components. This is why the change in the number of occurrences of these system calls is generally related with the user idleness. Hence, to build the training set, we consider as standard vectors those with a low number of occurrences of these system calls and the user idle, and those where the number of occurrences is high and the user is active.

In addition to the previous 12 features (11 system calls and user idleness), the vectors used for classification also includes a further feature representing the number of sent SMSes in the time interval T. In fact, monitoring SMS usage is semantically difficult through system calls only and SMS messages can be used to harm the user, stealing her credit. Moreover, SMSes are strongly related with the user activity. In fact, in a normal usage, an SMS is sent after the user has composed the message, which requires an active interaction. However, some applications send or receive SMSes to provide some kind of services. Since, SMS is a costly service, if compared to the amount of data that are sent with a message, applications should avoid SMS as communication channel as much as possible, and they should require that the user actively agrees with the sending of each message. Applications that send SMS messages when the user is idle should be considered suspicious. For all these reasons, we have logged several SMS sending phases, which represent real-life usage scenarios, and we have added the resulting vectors to the dataset.

Classifiers are not able to recognize a suspicious element if they are not trained also with some elements that belong to the suspicious behavior class. As previously said, a suspicious behavior is one that strongly deviates from those known to be good. Hence, we have manually defined some suspicious elements by creating both vectors with a high number of system call occurrences, when the user is idle, and vectors with an extremely high number of system call invocations, when the user is active. Figure 2 depicts some examples of standard and suspicious vectors. The picture depicts four sample vectors monitoring occurrences for 11 system calls with $T = 1$ sec. The last number of each vector means 1 for user active and 0 for user idle.

[2] For a full list of tested applications see `http://www.iit.cnr.it/node/15102`

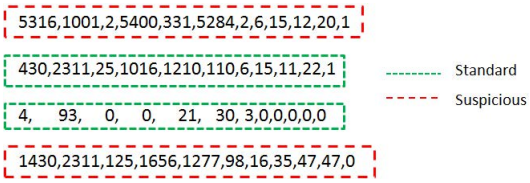

Fig. 2. Sample Vectors Monitoring Occurrences for System Calls and Idleness

More malicious vectors were derived from the ones that we have defined using a data balancing method named *SMOTE* (Simple Minority Oversampling TEchnique), which creates new vectors from those provided by means of interpolation. To represent malicious behaviors concerning SMS messages, we have manually defined and added to the training set some vectors with a number of sent messages that is very high compared to the user activity. We would like to point out that if classifiers are trained using such a dataset, which does not include malicious vectors generated by real malware, then each malware, if detected, can be considered as a *zero-day-attack*.

To increase the detection rate, our application runs in parallel two instances of the same detection framework, with a different sampling period T. The first instance is a short-term monitor with $T_{short} = 1$ sec, whereas the second instance constitutes a long-term monitor with $T_{long} = 60$ sec (both values are configurable at run-time). The cooperation of these two instances detects different types of misbehaviors. The short-term monitor is more effective in detecting "spiky" misbehaviors, i.e. with sudden, brief and sharp increase of the system call occurrences. On the other hand, the long-term monitor is aimed at detecting misbehaviors that distribute their action constantly in a long period of time, such as spyware, i.e. whose effect is not immediate.

Hence, two different datasets were built and used to train two classifiers of the same type. The classifier is a K-Nearest Neighbors (K-NN) [17] with $K = 1$ (1-NN). This classifier has very good performance and can easily adapt to a large number of problems, requiring a small amount of computation time to classify an element and a trivial update algorithm. We have also tested several other classifiers on our dataset but the 1-NN outperforms them all.

4.2 Experiments Description

Figure 3 describes at a high level the sequence of steps performed during the experiments.

During the *Training Phase*, the classifiers are trained with the initial, manually-defined, training set described in Sect. 4.1. The *Learning Phase* follows the training phase and it is used to learn behaviors that are specific of the user. This phase has been used to obtain an estimate of the False Positive Rate (FPR) trend (see Sect. 4.3), i.e. how the number of false positives decreases as they are used to progressively update the trained classifiers. During the *Operative Phase*, the trained

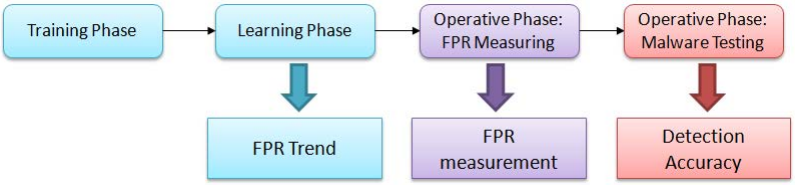

Fig. 3. Experimental Phases

classifiers are used to perform anomaly detection. During this experiment, this phase has been divided into two sub-phases: during *FPR measuring* the device has been tested with clean applications to compute the number of false positives raised per day; in *Malware Testing* trojanized applications have been installed on the device to determine the detection accuracy of the MADAM classifiers. Since the learning phase and FPR measurement greatly depend on the usage of the device, these tests were performed by three distinct users.

The next two sections describe the tests performed during the *Operative Phase*.

4.3 False Positive Measurement

Anomaly-based IDS have been criticized since they are more likely to generate false positives. False positives may strongly reduce the device usability, so we have performed a critical analysis of their occurrence on our system.

FPR Trend. A first experiment has been performed to estimate the FPR trend, i.e. how the number of false positives decreases as they are used to progressively update the trained classifiers. The training set that we have manually defined and given to the short-term classifier contained 900 standard vectors and 100 malicious ones. The long-term classifier has been trained with 250 standard vectors and 50 malicious ones. These datasets are relatively small and they represent some standard and basic behaviors (for the standard vectors), such as phone calling, SMS messages typing and sending, Internet browsing and gameplay of the popular game Angry Birds. Soon after the first dataset had been manually set up, and the classifier started, as we expected some false positives were raised (see Tab. 1 for details). False positives are likely to occur when the user performs a new behavior that strongly differs from those stored in the training set. Due to both the high number and the diversity of applications available for Android, unknown behaviors are likely to occur.

To reduce the occurrence of false positives, MADAM has to learn how the user behaves in an initialization phase, which we call the *learning phase*, where false positives are directly added to the classifier knowledge base without any user intervention. During the tests, the average duration of a learning phase to obtain a reasonable number of false positives is 30 minutes. However a new learning phase can be initiated actively by the user when she wants to update the classifier

with the generated false positives, for example by a newly installed application (if she considers that application trustworthy). During this experiment we have updated the classifier in five steps: after ten minutes and then each hour for four hours. More details on this experiment are reported in Tab. 1.

Table 1. Learning Phase

Time	10 min	60 min	120 min	180 min	240 min
Vectors	610	3050	3660	3660	3660
False Positives	156	55	23	10	5
FPR	0.26	0.015	0.0061	0.0028	0.0011

FPR Measurement. We used the training set obtained from this learning phase to re-train the classifiers, and then we performed furthers experiments to estimate the number of false positives raised in 24 hours. During these tests, one of the smartphones has been equipped with more than 50 applications and heavily used during the day. The other two smartphones have been set up, respectively, for moderate and basic usage. As expected, these smartphones with the lowest usage have raised a lower number of FP than the first one[3]. For this reason, here we only focus on the tests performed on the heavily-used smartphone.

The test returned 15 false positives ($FPR = 0.000171$), 9 of which were raised by the short-term classifier ($FPR = 0.000104$) and 6 by the long-term one ($FPR = 0.004167$). We updated the classifiers with the collected false positives, and then we reiterated the experiment for the following two days. Table 2 shows these results: on the average, on the heavily-used smartphone less than ten false positives are raised during 24 hours, with a descending trend.

Further tests have been performed using the non-trojanized version of some applications used for malware detection, to check if they would raise false positives as well[4]. We installed a clean version of the web browser `Opera` and of the `Hamster Bomb` game, while their trojanized versions were infected respectively by `OpFakeA` and `TGLoader`. As expected, no intrusions were detected.

Table 2. False Positive Rate

Day	**Overall FPR**	T_{short} **FPR**	T_{long} **FPR**
1	0.000171	0.000104	0.004167
2	0,000139	0.000116	0.00137
3	0.000114	0.00008102	0.00208

[3] During these experiment the classifiers have not been updated with the false positives, which were added to the training set only at the end of the experiments.

[4] These tests have been performed on some applications only because of the difficulty of finding a clean and trustworthy version of all the trojanized applications, which are only available on un-official markets.

4.4 Malware Detection

We have tested MADAM with real Android malware hidden in trojanized applications: all the malware applications are taken from a repository[5] that is updated as soon as new threats are discovered. The tested malware belong to different categories, e.g. Trojan, Rootkit and Spyware.

Table 3. One of the Malicious Vectors Monitored of `OpFakeA` Malware

open	ioctl	brk	read	write	exit	close	sendto	sendmsg	recvfrom	recvmsg	idleness	SMS Num
2246	25481	4341	47	16899	14416	12916	178	139	179	186	0	2

Each malware has been monitored as standalone to avoid cross malware detections. Furthermore, to reduce the likelihood that the suspicious vector has been caused by a false positive, each malware has been tested three times, restoring the device to a clean state after each test. Table 3 reports one of the vectors that MADAM (the T_{long} instance) classified as malicious during the infection of the malware `OpFakeA` (see Tab. 4). The last two elements of the vector are the most important: they mean that 2 SMS messages sending requests have been issued in a time interval with no user activity, a behavior that should be considered malicious for the SMS policies formerly discussed. After these tests, the dataset has not been updated with the suspicious vectors, so that each detected malware can be effectively considered as a zero-day-attack.

Table 4 shows the results of the three tests performed for each malware. The table also specifies which instance of the system monitor, i.e. short-term (T_{short}) or long-term (T_{long}), has detected the malware. This should give an idea of which type of misbehavior is performed by the malware. Those malware that have been detected by both classifiers are usually the most aggressive. We will further discuss these results in Section 5.

4.5 Performance

In the performed tests, the MADAM's impact on performance has not greatly influenced the user experience. The users have not noticed any reduction in responsiveness or in general visual performance. The periodic services of MADAM require an average of 7% of CPU overhead and of 3% MB of RAM space. The native battery monitor of the Android settings reports that MADAM uses only 2-5% of the total smartphone battery.

5 Discussion

To the best of our knowledge, MADAM is the first *real-time* anomaly-based malware detector developed for *real devices*, specific for Android, that is able

[5] `http://contagiominidump.blogspot.it/`

Table 4. Malware Detection Results

Malware	Type	Detection Rate	T	Description
`Lena.B`	BootKit	100%	T_{short}	Modifies files in the system partition.
`Moghava`	Trojan	100%	T_{long}	Modifies pictures stored on the device. Gradually fills the SDCard memory.
`TGLoader`	RootKit	100%	Both	Obtains root privileges, installs other malicious applications, opens a backdoor.
`OpFakeA`	Trojan	100%	T_{long}	Sends SMS with SIM data, downloads applications and stores them on the SDcard.
`NickySpyB`	Spyware	66%	Both	Record calls, stores them on the SDcard then sends them with other user's data to an external server.
`Gone in 60 sec`	Spyware	66%	T_{short}	Sends user's data to an external server.
`KMin`	Trojan	100%	T_{long}	Sends SMS to premium rate numbers
`Lotoor`	Rootkit	100%	Both	Obtains root privileges and opens several backdoors.
`DroidDream`	Rootkit	100%	T_{long}	Obtains root privileges and opens a backdoor.
`Droid Kung Fu`	Rootkit	100%	Both	Sends device information to a remote server.

to detect *real malware* of different categories. The detection results and FPR are better than those of previous anomaly-based detection systems for Android ([4], [2]).

The overall detection accuracy was of 100% for all malware, except for two that were not detected in one of the tests. These malware (`NickySpy` and `Gone in 60 seconds`) are spyware and their behavior is less aggressive compared to that of the other monitored malware. `NickySpy` records all the calls on the SDCard and then sends them via `HTTP` to an external server. The monitored device behaviors during a normal call and during an eavesdropped one show clear differences. However this misbehavior can be confused with the one in which a heavy application, such a 3D videogame, is running and, hence, the system can be deceived in such a situation.

`Gone in 60 seconds` is not a real malware but an application that a user intentionally installs on the device and when it is started reads all the user data, such as SMS messages and contacts, and sends them all to an external server. Then, the applications is automatically uninstalled, all in no more than 60 seconds (hence the name). During the execution, the application displays to the user a number that can be inserted on a website, hosted on the same server where the data have been sent, to retrieve the data. The behavior of this malware results more aggressive when there are much more data on the phone, in the other cases its detection can result tricky.

It is important to underline that, differently from previous approaches, our proposed framework is not based upon a per-application monitoring: instead, it performs a *global monitoring*, i.e. it is oblivious of which application(s) generated the event(s). This method can be more effective in identifying sudden behavior changes: as an example, a method that could be used to trick per-application controls is that of developing some applications, harmless if taken as standalone, but that can cooperate to perform an attack. A proof-of-concept of these applications is presented in [4], where the malicious application, a video-game, looks

harmless because it does not ask any dangerous Android permission. However, after the installation, this application shares the permissions with another trojanized application that does not performs malicious operations, but that has the permission to send both SMS and MMS. Then, the video-game starts to send sensitive information about user's contacts by means of SMS messages. Such an attack should be identified more easily by means of a global monitoring system, which considers all of the system calls issued in a time interval. Being a global anomaly detector, MADAM is able to detect an intrusion attempt but it is not able to detect the malicious source. However, its response can be used to trigger further components able to track and stop the source of the malicious behavior.

A question that may arise is how the user is able to distinguish between a false positive and a real intrusion. After the learning phase, occasional false positives become a rare event, so occasionally detection can be related with them. In fact, all the tested malware, show aggressive behaviors that cause periodical and multiple detections in a limited period of time. The only exception concerns SMS-based malware, which should be handled with an *ad hoc* strategy. A possible extension to this framework can automatically handle the occasional false positives, or can guide the user through a smart learning phase, to learn as much as possible from her behavior in a short period of time. The same framework can be used to trigger a new learning phase when a new trustworthy application is installed.

6 Conclusions and Future Works

In this paper, we have presented MADAM, a framework that allows early detection of intrusion attempts and malicious actions performed by real malware for Android devices. The framework exploits a multi-level approach, i.e. that combines features at the kernel-level and at the application level, and is based upon machine learning techniques. The first set of results is encouraging: the first prototype of MADAM for Android smartphone has managed to detect all the 10 monitored real malware, with a negligible impact on the user experience due to the few false positives issued per day. To the best of our knowledge, these results are a noticeable improvement to solutions presented in previous work, both for detection rate of real malware on current Android-based smartphones, and occurrences of false positives.

Since the tests provided promising results, we are working on an extension of this framework that combines the global monitoring approach with more specific monitors that consider additional features. With this extension we would like to create a database of expected behaviors that are related to high level actions, suck as starting a phone call. This should increase the system accuracy and allow the detection of a larger number of malware. Finally, whenever an alarm is triggered by this architecture, a further extension requires the tracing of the running applications to detect and stop the source of the attack.

References

1. Juniper Networks: 2011 Mobile Threats Report (February 2012)
2. Burguera, I., U.Z., Nadijm-Tehrani, S.: Crowdroid: Behavior-Based Malware Detection System for Android. In: SPSM 2011. ACM (October 2011)
3. Mutz, D., Valeur, F., Vigna, G.: Anomalous System Call Detection. ACM Transactions on Information and System Security 9(1), 61–93 (2006)
4. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y.: Andromaly: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38(1), 161–190 (2011)
5. Damopoulos, D., Menesidou, S.A., Kambourakis, G., Papadaki, M., Clarke, N., Gritzalis, S.: Evaluation of Anomaly-Based IDS for Mobile Devices Using Machine Learning Classifiers. Security and Communications Networks 5(00), 1–9 (2011)
6. Bose, A., Shin, K.G.: Proactive Security For Mobile Messaging Networks. In: WiSe 2006 (September 2006)
7. Jacoby, G.A., Marchany, R., Davis IV, N.J.: How Mobile Host Batteries Can Improve Network Security. IEEE Security and Privacy 4, 40–49 (2006)
8. Schmidt, A.-D., Peters, F., Lamour, F., Scheel, C., Çamtepe, S.A., Albayrak, S.: Monitoring smartphones for anomaly detection. Mob. Netw. Appl. 14(1), 92–106 (2009)
9. Xie, L., Zhang, X., Seifert, J.-P., Zhu, S.: pBMDS: a behavior-based malware detection system for cellphone devices. In: Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24, pp. 37–48. ACM (2010)
10. Bose, A., Shin, K.G.: Proactive security for mobile messaging networks. In: WiSe 2006: Proceedings of the 5th ACM Workshop on Wireless Security, New York, NY, USA, pp. 95–104. ACM (2006)
11. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: CCS 2009: Proceedings of the 16th ACM Conference on Computer and Communications Security, New York, NY, USA, pp. 235–245. ACM (2009)
12. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically Rich Application-Centric Security in Android. In: Annual Computer Security Applications Conference, ACSAC 2009. pp. 340–349 (December 2009)
13. Schmidt, A.-D., Bye, R., Schmidt, H.-G., Clausen, J.H., Kiraz, O., Yüksel, K.A., Çamtepe, S.A., Albayrak, S.: Static Analysis of Executables for Collaborative Malware Detection on Android. In: Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, June 14-18, pp. 1–5. IEEE (2009)
14. La Polla, M., Martinelli, F., Sgandurra, D.: A survey on security for mobile devices. IEEE Communications Surveys Tutorials (99), 1–26 (2012)
15. Kwak, N., Choi, C.H.: Input Feature Selection for Classification Problems. IEEE Transactions on Neural Networks 13(1), 143–159 (2002)
16. Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., Estrin, D.: Diversity in Smartphone Usage. In: MobiSys 2010. ACM (June 2010)
17. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory IT-13(1), 21–27 (1967)

Using Low-Level Dynamic Attributes for Malware Detection Based on Data Mining Methods

Dmitry Komashinskiy and Igor Kotenko

St. Petersburg Institute for Informatics and Automation (SPIIRAS)
39, 14 Linija, St. Petersburg, Russia
{komashinskiy,ivkote}@comsec.spb.ru

Abstract. The modern methodologies of computer threats' detection traditionally include heuristic approaches of detecting malicious programs (malware) and their side effects. Usually these approaches are used in order to form some auxiliary classification and categorization systems which simplify procedures of processing previously unseen data sets and revealing previously non-obvious structural and behavioral dependencies for malware. Such systems have a number of issues caused by specificity of processes of their creation and functioning. One of such issues is looking for feature sets whose use increases accuracy of malware detection. The paper presents description and analysis of an approach focusing on this issue. It is based on instantiating a number of classifiers learned in a feature space representing low-level dynamic specificities of applications to be analyzed.

Keywords: malware detection, data mining, dynamic attributes.

1 Introduction

The one of the most serious challenges in the information security domain is the task to timely detect and counteract malicious software (malware). Roughly speaking, the phenomenon of malware appeared first time twenty five years ago and nowadays it posed a big problem for the whole informational society. We do not plan to go through the detailed description on this statement and just want to mention several numbers. At the very beginning of 2000 years research community and antivirus vendors registered about 10 new instances [1] of malware each day. At this moment this value is equal to several tens of thousands. In accordance with the public statistical data of antivirus vendors [2] it is possible to perform an extrapolation and assume that the total number of registered malware samples will approach important threshold expressed by 100 million by the end of 2012 year. There is yet another important aspect which must be taken into account. A couple of years ago the malware technologies were taken into use as armor for future local and global conflicts. There are two well-known incidents related to Stuxnet [3] malware used for attacking some industrial units and Flamer [4] case which can be considered as an implementation of universal platform supporting total espionage and other potentially dangerous activities. We

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 254–269, 2012.

agree that anti-malware technologies are being developed and maintained very well by responsible parties; however it is also visible that as a whole the malware problem is far from its complete decision. Most likely this problem will remain with us next decades and it is understandable that the malware detection and counteraction issue is a science-capacious domain requiring applying of multi-model analysis approach.

The use of heuristic systems searching for malware is a one of the approaches to prepare some practical solutions for its generic detection. As a rule, the essence of these systems is based on extracting and formalizing some finite set of patterns which are specific to some particular kinds of threats. At their initial phase of evolving, such heuristic rules were formed manually and generalized some knowledge domains. However, soon, with the growth of samples' amounts and complexity, the task of comprehensive manual analysis of each sample became more and more expensive. This stipulated the interest of research community to alternative methods on data analysis, including Data Mining approaches. Being initially formulated for the malware detection task in the middle of 1990 years by Kephart et al. [5], this direction then got substantial impulse for comprehensive development by attempts to map already existing successful spam detection approaches to malware domain at the middle of the first decade of 21-st century.

The global problem of heuristic malware detection approaches is in their insufficient accuracy, expressed in so-called False Positive (benign object is treated as malicious) and False Negative (and vice versa) errors. This stipulates the main function of these methods – basically they play a role of auxiliary automated means providing malware experts with some kind of initial filters of incoming data stream and allowing them to focus on questionable and suspicious objects.

The gradual realizing by society of the fact that the protection of information resources is crucial and new promising technologies like cloud-based services are developed mitigate the problem of False Positives a bit. This causes an assumption that in future the importance of heuristic malware detection systems will increase, while the accuracy reserve exists and can be used in order to improve the state of False Negative issues.

This assumption can be confirmed by the fact that last years a lot of auxiliary systems of objects' and their origins' validation and reputation scoring are introduced and widely used by antivirus community and advanced users. Such measures allow users to identify and decide on questionable situation on their own; however it does not exclude the need to continue works on further improvements of the good old heuristic malware detections.

There are many approaches to extract features reflecting different structural and behavioral characteristics of software applications. They are adopted successfully to reveal some aspects of malicious applications and can be used for constructing detection systems both for already known and for previously unseen malicious files. Their success is stipulated by one important factor. It is known fact that nowadays malware is a massive, high-spread phenomenon causing appearance of tens thousands of new malicious samples per day. Such huge volume of new data could not be reached without the use of automated environments adopting protection techniques focused on preparing new unique binary objects those analysis must be extremely difficult. However, these

protection techniques do not evolve so fast and they have own structural and behavioral specificities which become visible when the malware counteracting party continuously analyzes big data blocks. This visibility makes generic detection of modern malware possible when it comes to Data Mining. Therefore modern heuristic malware detection systems are rather oriented on the search for protection patterns specific for malicious objects than on looking for malicious behavior. These patterns potentially can be observed both statically and dynamically.

The review of feature extraction approaches shows that the issue of using low-level dynamic features based on instructions' sequences was not elaborated (for instance, in contrast to n-grams based approaches) and has significant potential to appear useful for malware detection. The paper describes and analyses an approach to implement qualitative heuristic system of malware detection based on Data Mining methods. The essence of the approach is in using additional method to extract dynamic features which can be adopted by malware detection systems. In particular, we investigate an issue of extracting behavioral (blocks of machine code instructions) attributes of malware delivered in portable executable format.

The paper's structure is as follows. "Related Works" section provides a brief description of activities done by the research community in the context of applying Data Mining techniques for constructing malware detection and categorization systems. In the section we mainly focus our attention on used features. The next "Approach description" section defines a generic application model we use, describes its instantiation for our research and explains the approach we use for extracting feature space items. Then "Experiments" section covers tools, sequence of practical steps we performed, and presents their results. "Discussion" section provides analysis of the results, discusses visible advantages and disadvantages of the approach we present and suggests some ways how to mitigate the latter ones. The paper ends with "Conclusion" part emphasizing main points of the research.

2 Related Work

The real interest of the research community to the issues of using Data Mining methods appeared first time at the very beginning of the 21-st century. Initially its appearance was stipulated by the need in the already mentioned systems of automatic classification and categorization of the permanently increasing stream of new, previously unseen executable binaries and other file types. At the moment of realizing this need the research community already had successful patterns of using Data Mining techniques for detection co-called spam (junk) e-mails. This fact caused the beginning of active research in the alternative, malware-specific domain.

However, it is important to pay attention to the fact that initially the methodology of using these methods for malware detection was described in the middle of 1990th in the research work of Kephart et al. [5] for Portable Executable format [6].

While looking for the best practices in the scope of this methodology it becomes clear soon that the success of any approach to use Data Mining methods for malware detection is defined by two main factors.

- Such research must have enough amounts of well-formed malware samples (other data items) which are to be used for forming learning and validation data sets. Nowadays the most successful decision for the former (data availability) factor is to use well recommended in research work data sets which are available publicly, for instance [7] or data collected by antivirus vendors and laboratories. Thus, the first issue is not too problematic at this moment in our point of view and other discussion of the topic is out of the paper's boundaries.
- There is the need to choose a right way to extract features used for preparation generic vector describing the objects of experimental data set (feature space). This factor (feature extraction approach) is important at this moment and still actively being discussed in this knowledge domain. Thus paper is devoted to this question and further part of this section is devoted to description and brief discussion of the already known features of file objects and their advantages and disadvantages.

Fig. 1 presents the high-level structure of the whole features applicable for detection of malware spreading in the form of separate file objects (files). It can be treated as an aggregate of two distinctive feature groups. External features in respect to a file object include, for example, WhoIs Web service data about file's origin URI, diverse services and catalogs keeping opinions on the object from user (for example, Facebook) and expert (for instance, VirusTotal) communities etc. Internal features' group of an object consists of data extractable exceptionally from the object during its processing phase.

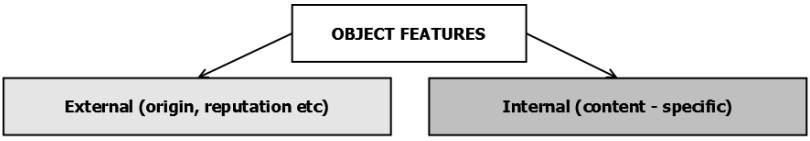

Fig. 1. High-level separation of file objects' features

Fig. 2 shows detailed representation of internal features' group structure. The features are used in diverse malware detection methods, especially for PE32 file objects analysis. It can be seen than the main criterion of separating the space of internal features is defined by methods of their extraction, defining types of the detection approaches (static and dynamic). The static group of methods is based on objects' analysis without context of its processing by an interpretation environment and includes all statically extractable features while the dynamic group of detection methods uses behavioral information about objects' interaction with the environment presented in form of so-called dynamic features.

There are different static features. They can be classified into a number of generic groups:

- N-grams, or fixed-length byte sequences;
- Strings, or symbol sequences, whose content corresponds to a number of predefined conditions;

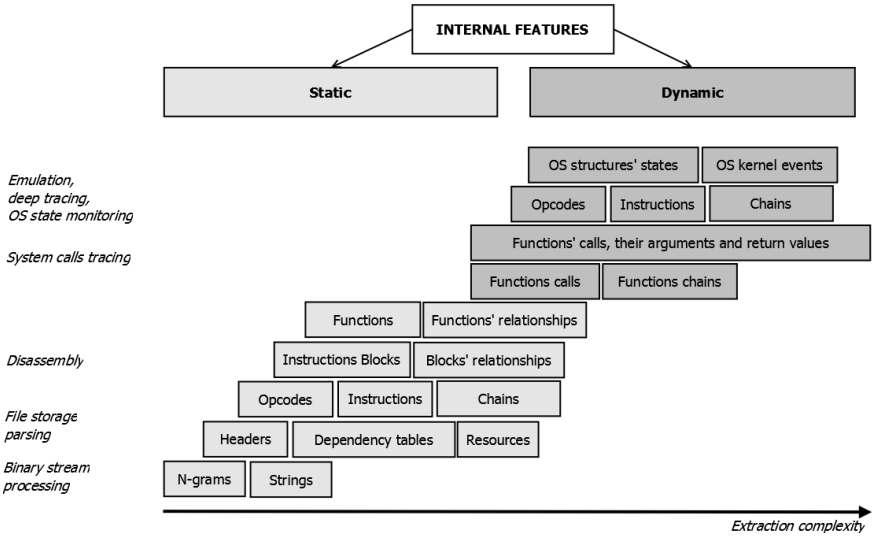

Fig. 2. Detailed representation of internal features groups

- File headers content, including subgroups of generic headers, module dependencies' tables, resources and other data types;
- Opcode-level information about machine code located in a file object to be analyzed;
- Data about instruction sequences (opcode, argument-level) included in a file object;
- More sophisticated, high-level data on static blocks of instructions (translation blocks, instructions' chains, etc);
- Information about logically separated sets of instructions' blocks (e.g. separated functions);
- Data about interdependencies of separated blocks of machine code (translation blocks, functions, etc).

The first research where n-grams were applied in order to extract features of malicious code is the work of Kephart et al. [5], devoted to important at that time topic on detection infected boot sectors. The one of the fundamental researches dedicated to efficacy of n-grams inspired approach to detect malicious PE32 files is the work of Shultz et al. [1], where the authors applied the technique for extracting short byte sequences with length equal 2 bytes. Further n-grams efficacy was touched many times by other research groups. Kolter and Maloof [8] continue investigation on n-grams as features for detection malicious PE32 files. They define n-grams lengths in range from 1 to 10 and extract them from code sections of PE32 files. They report that optimal accuracy for n-gram based malware detection models is observed with n-gram length equal to 4 and feature set size equal to 500. Masud et al. [9-11] actively use n-grams to build generic approach for malware detection by feature combining. Menahem et al. [12] adopt this feature group for evaluating their hierarchical approach to combine classification

methods for malware detection. Alazab et al. [13] and Santos et al. [14] use n-grams to continue experiments with building heuristic malware detection systems. The main advantages of the n-grams as features are their visibility and simplicity to extract. However, this group of features has significant disadvantages defined by their potentially huge number that makes their processing very resource-consuming task and, correspondingly, makes further feature selection steps problematic. Moreover, n-grams are not applicable when it comes to a work with protected (packed, obfuscated) file objects whose analysis is very difficult and requires removing of code / data protection levels.

Strings, also referred to as interpretable (readable) strings, can be considered as a group of features which is semantically close to the aforementioned byte sequences (n-grams). Their specificities are (1) variable lengths; (2) their symbols must conform to some predefined diapason of values defining string alphabet and (3) availability of terminal symbol indicating end of a string. Already mentioned research of Shultz et al. [1] can be treated as a one of the first works which use string data as a source of features for building malware detection system based on Data Mining methods. The similar idea of using strings is adopted in signature-based malware detection by antivirus vendors. Kolter et al. [8] show that often the most valuable n-gram features for malware detection process correspond to symbols sequences belonging to strings. Nowadays due to some disadvantages the static string features are less significant, but anyways they are still paid some attention. E.g., Lu et al. [15] use the string features as one of the features' groups for preparing combined classifiers scheme for malware detection. The main advantages of string features are the same we mentioned for n-grams plus theirs very high interpretability. However they inherit the same disadvantages we mentioned for n-grams.

Headers' data of binary executables is substantial source of information for heuristic malware detection systems. Shultz et al. [1] apply the data of import tables to prepare features' subset which informs about user-mode system functions used by applications. Menahem et al. [12] use general PE32 headers' values and import, export and resources tables' data in order to build multi-layer hierarchical classification system with combining different features types. Perdisci et al. [16] take headers' data to extract a set of derivative features (entropy, data of executable PE32 sections and so forth). Shahzad et al. [17] perform analysis of header ELF structures representing binaries' structure for UNIX-originated operating systems. They show that using semantically similar structural data for different platforms can provide some heuristic detection mechanisms. The main advantages of this feature group are relative simplicity of their extraction and interpretability. The proper usage of this data type allows to obtain an initial state of an application being stored by an executable binary that in some sense defines its further functionality. However the availability of advanced protection techniques to hide real initial state of the process doesn't allow applying this feature group advantage comprehensively.

The groups of features extractable from an application (executable binary) at disassembly level allow obtaining low-level information characterizing its static "behavioral" aspects in the form of CPU instructions and theirs blocks (sequences, functions). Ye and Lee [18] use instructions' appearance frequencies and static sequences of instructions at functions' level. Masud et al. [9-11] adopt static chains of instructions'

representations which include semantic types of operations and arguments' types. Siddiqui et al. [19] suggest an approach to form instructions' sequences' descriptions as raw opcode values chains. Alazab et al. [13] use static data about called isolated functional code blocks of instructions (functions) in order to improve n-grams based malware detection system. Kinable [20] proposes a way to use static data on interdependencies of functional instruction blocks to prepare characterizing static code flow graph which is used then as a data for measuring similarity with other graphs extracted from other application. The groups of features are the most informative ones from the whole set of static data. Their main pros are relatively easy extraction procedures (which, for sure, more complex in comparison with n-grams extraction) and significant semantic meaning. The cons of the feature group are similar to other static feature groups – potentially huge amount of available data and shortcomings related to impossibility to counteract typical code protection mechanisms.

Dynamic features used in Data Mining – inspired approaches of malware detection are listed as follows:

- Information about external code used by an application during its execution (usually API functions, in particular – functions and services of Operating System);
- Data about sequences of used external code (e.g. sequences of API calls);
- Information about input arguments for called functions and theirs return values;
- Low-level data on executed instructions – opcodes, operands, sequences etc;
- Data about changes of internal kernel structures of Operating System and kernel events.

The one of our previous research works presents an approach to detect malware by monitoring system kernel services being used by analyzed application for Windows XP operating system [21]. Feature space was represented by numerical and nominal features characterizing, for example, amount of particular calls and access type to some specific systems resources correspondingly. In contrast to this research, Ye et al. propose a classic way to use sequences of monitored API calls' sequence as a source of features for malware detection systems. Lanzi et al. [23] use so called "behavioral n-grams" for forming feature space which is prepared by passing through the system services call sequence of sliding window. The authors showed that the most efficient lengths of the sliding window are 4, 3 and 2 (in decreasing order). It is notable, that the call sequences were formed in accordance with system-centric approach oriented on access monitoring to crucial system resources (files, registry keys and values etc). Rieck et al. [24] adopt chains of symbols identifying semantics of monitored system calls and their arguments. The feature extraction approach was based on sliding window method as well. Shahzad et al. [25] present approach to detect malware by monitoring kernel mode structures' state and value associated with an application (internal process structures) being monitored. The approach is applied for UNIX-originated operating systems and in theory can be used for PE32 format and, correspondingly, for Microsoft Windows as well. The author stated that it is enough to monitor kernel structures at the very beginning of process's life cycle to decide whether it is malware or not.

3 Approach Description

The essence of the approach we suggest is based on the representing of any executable binary object (software application or shared library) as a sequence (or set of sequences for multithreaded applications) of CPU instructions being executed during application's functioning. The task of sequence analysis for Data Mining approaches usually requires some formal simplification method which is able to convert sequences (objects' representations) to a finite set of features those values could characterize any sequence (object), e.g. mentioned in "Related works" section so-named sliding window approach. Thus in order to proceed with the next level of the approach description we have to introduce a formal model of software application, its instantiation parameters and some basic practicalities on the feature space preparation process.

The formal model M of a software application can be stated as following.

Let's introduce *alphabet* A as a finite set of symbols $a_1, \dots, a_k : A = \{a_1, \dots, a_k\}$, where $k = |A|$ – alphabet's size and $a_i \in A, 1 \le i \le k$.

Execution thread T is determined as an ordered finite set of symbols $(a_1, \dots, a_n)$ of alphabet A with size $n, n \in \mathbb{N}$: $T = (a_1, \dots, a_n)$, $a_j \in A, 1 \le j \le n$.

An *application* P is defined as a finite set of execution threads $\{T_1, T_2, \dots, T_m\}, m \in \mathbb{N}$: $P = \{T_1, T_2, \dots, T_m\}$.

Let's define a *set of terminal symbols* A^T as a subset $\{a_1^T, \dots, a_t^T\}$ of all set of alphabet A symbols, $a_l^T \in A, 1 \le l \le t$: $A^T \subset A$.

The concept of *terminal symbols' chain* C is defined as an ordered finite set of alphabet A symbols of size $l, C = (a_1, a_2, \dots, a_l)$, where only first and last symbols belong to the set of terminal symbols A^T:

$$chain(\mathrm{C}) = \begin{cases} c_1 \in A^T \\ c_l \in A^T \\ (\forall i, 2 \le i \le l-1)(c_i \in A, c_i \notin A^T) \end{cases} \quad (1)$$

Then with some minor assumptions an *execution thread* T can be represented as a sequence of terminal symbol chain: $T = (C_1, C_2, \dots, C_l)$, $(\forall i)(chain(C_i))$.

The *model of application* will is determined as a set $M = (P, A, A^T)$, $A^T \in A$, including collection P of execution threads T of size m, collection (set) of alphabet A symbols of size k and a subset of terminal symbols A^T of size t, $A^T \in A$:

$$M = (P, A, A^T), A^T \in A. \quad (2)$$

Having the formal application model specified lets define terms of a *feature* (*attribute*), *feature-based object description* and *feature space*.

A *feature* is a function $p: O \to V_p$, where V_p is defined as a finite set of permissible values of description and O is a set of objects.

Assume there is a set of features: $p_1, p_2, \dots, p_n$. Then the vector $d_o = (p_1(o), p_2(o), \dots, p_n(o))$ is a *feature-based description* of an object $o \in O$.

Then the set $X = V_{p_1} \times V_{p_2} \times \dots \times V_{p_n}$ is called as a *feature space.*

The task of malware detection can be characterized as follows. Assume there are a set of feature descriptions $D = \{d\}$ and a set of class labels $Y = \{y\}$.

There is an unknown target dependency – function $y' : D \to Y$, those values are known only for the objects belonging to the set of learning set $D^l = \{(d_1, y_1), \ldots, (d_l, y_l)\}$.

The task is to build algorithm (function) $a : D \to Y$, which is able to express maximal accurate conformance of an object $d \in D$ to a single element $y \in Y$ in accordance with the target function y'. Such task is usually solved with well-known set of Data Mining methods known as supervised learning (classification) approaches.

Now, after posing formal definitions we can proceed with their instantiation values. The approach suggested in the paper uses the model M of a software application a common software application abstraction providing where each terminal symbol chain is treated as a potential feature. This allows us to define feature space we use and build Data Mining-inspired detection model. The instantiation assumptions are following:

- *An application* P includes single *execution thread* T. Thus the size m of the execution threads' set P is 1. For the applied knowledge domain this assumption can be explained so, that the initial life cycle phase of any application includes only one, primary execution thread which is responsible for execution Entry Point code (startup code);
- *An alphabet* A is defined by instruction set of the CPU to be used for executing an application (binary object). In order to minimize the potential size of the alphabet it was decided to restrict symbol's size with 2 bytes. Thus, $k \leq 2^{16}$;
- *A set of terminal symbols* A^T is defined by a subset of CPU instructions which defines application's execution flow (so called conditional and unconditional control transfer instructions).
- All *features* have Boolean type: $V_p = \{true, false\}$. Each *feature* p indicates whether some particular chain of terminal symbols does exist in the object description while class label defines whether the objects is malicious or not (benign).
- Class labels set is defined as $Y = \{malware, benign\}$.

There is one important practical aspect specific for the suggested approach. The group of dynamic features we use is formed as a set of sequences of 16-bit size symbols characterizing executed CPU instructions. In accordance with Intel documentation [26,27], each instruction of IA-32 CPU is considered as a byte sequence including obligatory opcode with length $l \in [1,3]$. In order to instructions' layouts we use 16-bit representation of opcode, i.e. an opcode *op* of CPU instruction *instr* is a 2-byte structure with fields *op[1]* и *op[2]:*

- If *length(instr.opcode) == 1* then *op[1] = instr.opcode* and *op[2] = 0*;
- If *length(instr.opcode) == 1* and the opcode has so-called group extension [26] then *op[1] = instr.opcode* and *op[2]* = (*instr.modrm* & 00111000);
- Otherwise *op[1] = 0x0F*, *op[2] = instr.opcode*. The main reason of this simplification is relative rarity use of such instructions in traditional user-mode applications and opportunity to reveal their usage under such simplification circumstances.

The process of forming finite-length sequences of the symbols (terminal chains in terms of the formal model stated above) was organized with help of the rule of forming execution blocks restricted by conditional (for instance, JZ) and unconditional (for example, RET, JMP) CPU control flow instructions. An example of representing the typical code chain is shown in Fig. 3.

Fig. 3. An example of the dynamic feature representation process

We defined the following classification methods as base classifiers to be used in the experimental part of the work: Naive Bayesian Classifier (NB), Decision Tree (DT) and k Nearest Neighbors (kNN) [28]. The efficiency of using these classifiers is treated in context of their instances' accuracy for cases defined by different data sets and, therefore, different features' sets belonging to the feature group we discussed above.

4 Experiments

The experiments were performed with using the traditional data source [7] which is widely used in many researches on the malware detection topics [8-11]. From the whole dataset we picked a number of the following malware families: Bifrose [29], Lmir [30], Magania [31], OnlineGames [32], Poison [33] and Vapsup [34]. The set of benign files was extracted from several sources including sets of executable files from Microsoft Windows and popular Internet service SourceForge [35]. The total amount of files used during experiments was 13120 and 11958 malicious and benign files correspondingly.

The extraction procedure of the initial set of features was performed with help of several tools in semi-automatic way in specially prepared isolated environment. We used Debugging Tools for Windows [36] with its Python wrapper package PyDbg [37] and interactive disassembler IDA [38]. During the gathering of the data about instructions' chains we took into account the following restriction – the gathering process was oriented on the instructions, located in the virtual address space region where the object being analyzed was mapped in. This restriction excluded from consideration the common instructions chains belonging to the OS user-mode shared libraries. Moreover, the gathering process was limited by some additional conditions including time of analysis, amount of already available data and appearance of complex events (e.g. creating new threads, exceptions). These restrictions allowed to substantially simplify the gathering procedure logic, but made it possible to obtain only start-up instructions' sequences.

The obtained set of all available instruction chains' descriptions was divided into six separate subsets characterizing chosen malware families. In order to minimize calculations complexity we weighted by Information Gain Ratio [39] each such data subset and then selected top 500 features for the each one. Then these obtained shortened objects' descriptions were used for learning by chosen base classification methods. Rapid Miner 5.2 [40] was used in order to learn malware detection models, validate their accuracy and perform other necessary calculations. It includes both own means of supporting Data Mining – related computations and well-known Weka package which is traditionally used by research community for experiments on malware detection [8-11]. Fig. 4 depicts a fragment of the whole scheme of the experiment we did with help of this environment. Data loader item ("Read AML" element) passes obtained data to the data copy item ("Multiply" element). Then these copies are sent to blocks of target base models (elements "DT", "NB" and "KNN").

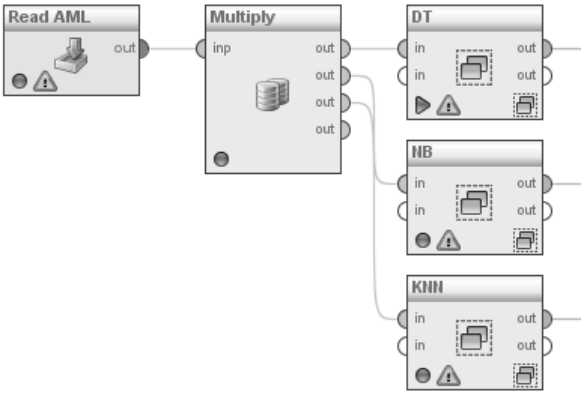

Fig. 4. A fragment of experiment scheme in RapidMiner's environment

Fig. 5 shows a fragment of decision tree built with use of data subset including Poison [33] and clean samples. The string data is decision tree's nodes which represent feature strings for instructions' opcode sequences. Decision tree's edges point decision making conditions. For example, condition $feature > 0.500$ corresponds to existence of the feature (opcode sequence) in the object's description.

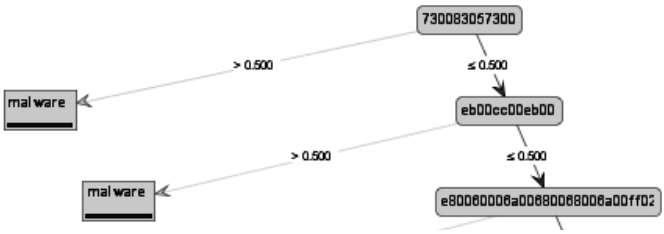

Fig. 5. A fragment of a decision tree built for Poison malware family

In accordance with the task posed for the experiments we obtained eighteen models of malware detection based on chosen base classification methods. Each of them was applied to all malware families' subsets. The obtained models' accuracy validation was performed with so-called 10-fold cross validation procedure. Matthews Correlation Coefficient [42] was used as a main accuracy indicator.

The obtained accuracy results are shown in Table 1 and Fig. 6.

Table 1. Results of experiments

	DT	KNN	NB
Bifrose	0.56971700	0.57091111	0.57269830
Lmir	0.46309339	0.47722767	0.47908319
Magania	0.72887724	0.73356166	0.74286805
OnlineGames	0.77920902	0.77905059	0.78047616
Poison	0.84038047	0.84271514	0.84736974
Vapsup	0.08217656	0.07928708	0.08633095

As it can be seen from the resulting table, in general case the most effective classification method is Naive Bayesian classifier. For proposed type of low-level dynamic features it can be adopted for instantiation some basic heuristic malware detection systems. However, it is visible that accuracy difference between the methods we used during experiments is not significant.

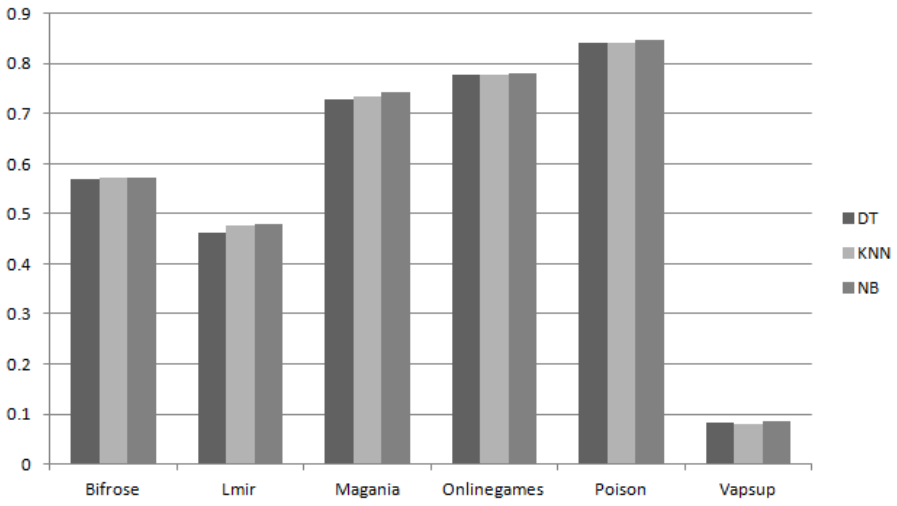

Fig. 6. Graphical representation of experiments' results

Fig. 6 shows that the proposed approach for dynamic features' extracting does not provide the same level of accuracy for different families of malware. It is most effective for Poison [33], OnlineGames [32] and Magania [31] malware families.

5 Discussion

The approach to extract behavioral (dynamic) features, evaluated in this paper, is based on representing of a Portable Executable file object in a form of a set of Boolean attributes. Each attribute characterizes availability in object's code of some specific CPU instruction sequence terminated by instructions of conditional or unconditional execution flow transfer.

We have to stress, that modern code or executable binaries generation tools are so diverse, that the approach cannot be considered as a "silver bullet" for detection of malicious programs.

However, in some specific cases depending on particular malware generators, custom obfuscators and protectors, the approach can be used with high efficiency in the context of the multi-model approach using a diverse set of features and data sets in order to prepare combined malware detection model.

The approach provides efficient means for revealing origin-specific aspects where under origin we assume some code generation tool (e.g. a unique protection tool) or a unique source code package. The imagination of an application as a set of instructions' chains adds new dimension of behavioral analysis requiring further deep investigation.

Moreover, the approach to extract dynamic features has some principal practical restrictions requiring more sophisticated implementation of raw data harvesting. In order to obtain a trace of instructions per each file we mainly used mentioned above debugging means. This causes harvesting environment's safety issues and a need to regularly revert it back to the initial state.

Usually malicious software uses a huge set of methods to detect whether an application is being debugged and to counteract debuggers that causes raw data quality related issues as well [43].

In our point of view, the first problem can be solved by using more safe and reliable (however more vulnerable to some specific tricks) PE32 emulation means.

The extent of importance of the second issue has not been specified very well yet and requires deeper look at problematic cases. However, in general case, while using other types of features identifying more high-level file objects' (applications) specificities, we can expect the issue's significance to decrease due to automatic involvement into the set of valuable potentially malicious attributes some patterns of preparing and performing such checks for debuggers.

During the thorough review of accuracy results for obtained models we realized that the absence of significant differences between classification models per each malware subset has the solid explanation. The overwhelming majority of features used by all classification methods is specific for malicious samples only. Thus, the used data sets including features extracted by proposed approach have a set of features that can be used to distinguish malicious and benign applications without building complex rules and models. This emphasizes strengths of the approach and reserves a place for its usage in the global multi-model malware detection schemes.

6 Conclusion

In the paper we have considered the issues of improving the quality for heuristic malware detection means based on applying Data Mining techniques. The approach presented in the paper specifies a way to represent low-level behavioral features which can be useful to distinguish different malware families and benign application of Portable Executable format (PE32). It can be adopted for other types of malicious executable binaries as well.

The idea of the approach is based on representing any application to be analyzed as a sequence of executed instructions. The sequence is considered as a main source of features (attributes) which are necessary to represent an analyzed application in vector form. In order to prepare the set of features, we suggested to defragment instructions' sequence into blocks of instructions bordered with control instructions (instructions changing execution flow). Instructions in our approach are presented by an alphabet consisting of 16-bit symbols including the IA32 instruction set specific data.

The approach to extract behavioral features which characterize applications' low-level dynamic specificities allows us to reveal some groups of patterns specific for malware and malware-intrinsic machine code protection tools (tricks). This fact makes the approach applicable in the multi-model generic malware detection.

The future work will comprise research activities focused on formal definition of the generic multi-model approach including already existing particular ways to detect malicious software with Data Mining tools and its practical validation steps.

Acknowledgments. This research is being supported by grant of the Russian Foundation of Basic Research (project #10-01-00826-a), Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (contract #3.2), State contract #11.519.11.4008 and partly funded by the EU as part of the SecFutur and MASSIF projects.

References

1. Schultz, M.G., Eskin, E., Zadok, E., Stolfo, S.J.: Data Mining Methods for Detection of New Malicious Executables. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 38–49 (2001)
2. McAfee Labs blog: A Look at One Day of Malware Samples (October 2011), `http://blogs.mcafee.com/mcafee-labs/a-look-at-one-day-of-malware-samples`
3. Wikipedia: Stuxnet computer worm, `http://en.wikipedia.org/wiki/Stuxnet`
4. Wikipedia: Flame computer malware, `http://en.wikipedia.org/wiki/Flame_(malware)`
5. Kephart, J.O., Sorkin, G.B., Arnold, W.C., Chess, D.M., Tesauro, G.J., White, S.R.: Biologically inspired defenses against computer viruses. In: Proceedings of 14th International Joint Conference on Artificial Intelligence, pp. 985–996 (1995)

6. Pietrek, M.: An In-Depth Look into the Win32 Portable Executable File Format. Microsoft Developers' Magazine (February, 2002), http://msdn.microsoft.com/en-us/magazine/cc135800.aspx
7. VX Heavens, http://vxheavens.com
8. Kolter, J.Z., Maloof, M.A.: Learning to Detect Malicious Executables in the Wild. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470–478 (2004)
9. Masud, M.M., Khan, L.R., Thuraisingham, B.M.: Feature-Based Techniques for Auto-Detection of Novel Email Worms. In: Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 205–216 (2007)
10. Masud, M.M., Khan, L.R., Thuraisingham, B.M.: A Hybrid Model to Detect Malicious Executables. In: Proceedings of the IEEE International Conference on Communication, pp. 1443–1448 (2007)
11. Masud, M.M., Khan, L.R., Thuraisingham, B.M.: A scalable multi-level feature-extraction technique to detect malicious executables. Information Systems Frontiers 10, 33–45 (2008)
12. Menahem, E., Shabtai, A., Rokach, L., Elovici, Y.: Improving Malware Detection by Applying Multi-Inducer Ensemble. Journal of Computational Statistics & Data Analysis 53(4), 1483–1494 (2009)
13. Alazab, M., Layton, R., Venkataraman, S., Watters, P.: Malware Detection Based on Structural and Behavioural Features of API Calls. In: Proceedings of International Cyber Resilience Conference, pp. 1–10 (2010)
14. Santos, I., Penya, Y.K., Devesa, J., Bringas, P.G.: N-grams-based File Signatures for Malware Detection. In: Proceedings of the 11th International Conference on Enterprise Information Systems, pp. 317–320 (2009)
15. Lu, Y.-B., Din, S.-C., Zheng, C.-F., Gao, B.-J.: Using Multi-Feature and Classifier Ensembles to Improve Malware Detection. Journal of Chung Cheng Institute of Technology 39(2), 57–72 (2010)
16. Perdisci, R., Lanzi, A., Lee, W.: McBoost: Boosting scalability in malware collection and analysis using statistical classification of executables. In: Proceedings of the Computer Security Applications Conference, pp. 301–310 (2008)
17. Shahzad, F., Farooq, M.: ELF-Miner: Using Structural Knowledge and Data Mining Methods to Detect New (Linux) Malicious Executables. Journal of Knowledge and Information Systems 30(3), 589–612 (2012)
18. Ye, Y., Li, T.: Automatic Malware Categorization Using Cluster Ensemble. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 95–104 (2010)
19. Siddiqui, M., Wang, M., Lee, J.: Detecting Internet Worms Using Data Mining Techniques. Journal of Systemics, Cybernetics and Informatics 6(6), 48–53 (2008)
20. Kinable, J.: Malware Detection through Call Graphs. Publications of Future Internet (FI) Programme, Master's Thesis. Aalto University, Department of Information and Computer Science (2010)
21. Komashinskiy, D.V., Kotenko, I.V.: Using Data Mining methods for malware detection. In: Information Fusion and Geographical Information Systems, pp. 343–359. Springer, Heidelberg (2009)
22. Ye, Y., Li, T., Huang, K., Jiang, Q., Chen, Y.: Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list. Journal of Intelligent Information Systems 35(1), 1–20 (2010)

23. Lanzi, A., Balzarotti, D., Kruegel, C., Christodorescu, M., Kirda, E.: AccessMiner: Using System-Centric Models for Malware Protection. In: Proceedings of the 17th ACM Conference on Computer and Communication Security, pp. 399–412 (2010)
24. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic Analysis of Malware Behavior using Machine Learning. Journal of Computer Security 19(4), 639–668 (2011)
25. Shahzad, F., Bhatti, S., Shahzad, M., Farooq, M.: In-Execution Malware Detection using Task Structures of Linux Processes. In: Proceedings of the IEEE International Conference on Communications ICC 2011, pp. 1–6 (2011)
26. Intel Corporation: IA-32 Intel Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M. Intel Corporation (2006)
27. Intel Corporation: IA-32 Intel Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, N-Z. Intel Corporation (2006)
28. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 161-168 (2006)
29. F-Secure: Bifrose malware family description, http://www.f-secure.com/v-descs/backdoor_w32_bifrose_bge.shtml
30. Total Malware Info: Lmir malware family description, http://www.totalmalwareinfo.com/rus/Trojan-PSW.Win32.Lmir.ko
31. F-Secure: Magania malware family description, http://www.f-secure.com/v-descs/trojan-psw_w32_magania.shtml
32. F-Secure: OnlineGames malware family description, http://www.f-secure.com/v-descs/trojan-psw_w32_onlinegames.shtml
33. Microsoft Security Portal: Poison malware family description, http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=TrojanDownloader:Win32/Poison.A
34. Microsoft Security Portal: Vapsup malware family description, http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?name=Adware%3aWin32%2fVapsup
35. SourceForge: Find, Create and Publish Open Source software for free, http://sourceforge.net
36. Microsoft: Download and Install Debugging Tools for Windows, http://msdn.microsoft.com/en-us/windows/hardware/gg463009.aspx
37. GitHub: Open RCE, pydbg, a pure-python win32 debugger interface, https://github.com/OpenRCE/pydbg
38. IDA: Interactive disassembler and debugger, http://www.idapro.ru/
39. Harris, E.: Information Gain Versus Gain Ratio: A Study of Split Method Biases. In: Online Proceedings of 7th International Symposium on Artificial Intelligence and Mathematics (2002)
40. I-Rapid: RapidMiner, http://rapid-i.com/content/view/181/190/
41. Weka 3: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/
42. Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta 405(2), 442–451 (1975)
43. Ferrie, P.: The Ultimate Anti-Debugging Reference (May 2011), http://pferrie.host22.com/papers/antidebug.pdf

Configuration-Based Approach to Embedded Device Security

Vasily Desnitsky, Igor Kotenko, and Andrey Chechulin

St. Petersburg Institute for Informatics and Automation (SPIIRAS),
39, 14 Linija, St. Petersburg, Russia
{desnitsky,ivkote,chechulin}@comsec.spb.ru

Abstract. Development of embedded devices is a challenging task because of their varying, reactive and real-time nature. Conventionally embedded devices are considered as a part of systems owned by some other entities and operated in a potentially hostile environment. Embedded device development is an extremely complicated problem due to various types of threats and attacks the device subject to, and because the security in embedded devices is commonly provided as an additional feature at the final stages of the development process, or even neglected. In this paper we propose a new configuration model, which facilitates the design of secure and resource consumption efficient embedded devices. The model enables the search for the most effective combinations of security building blocks in terms of consumption of device resources.

Keywords: embedded system security, security modeling, security building blocks, configuration, resource efficiency, non-functional property.

1 Introduction

From a security viewpoint, embedded devices are basically systems owned by a certain entity, used frequently as a part of systems owned by other entities and operated in a potentially hostile environment. Development of security-enhanced embedded devices is a complicated task owing to different types of threats and attacks that may affect the device, and because the security in embedded devices is commonly provided as an additional feature at the final stages of the development process, or even neglected. The paper encompasses the analysis of security issues of the systems which include embedded devices. Such systems are notable for autonomy of separate devices included in the system and for constrains of the resources of the device and their consequently weak efficiency [3, 7].

The approach proposed in the paper determines the way to combine particular algorithms and techniques implementing various protection properties. The goal of this approach is to achieve needed functional protection properties by choosing the most efficient combination of security building blocks, considering non-functional, resource-related properties of blocks and non-functional limitations of the device.

Importance of the objective of the paper is conditioned by occurrence and the tendency to rapid increase in quantity of devices carrying out communications on the

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 270–285, 2012.

Internet and directed remotely by wireless protocols – so-called "Internet of Things" [9]. Carrying out communications in non-controlled and potentially dangerous environment, such systems are exposed both to specified and universal attacks [10]. Hence, the task of building the efficient defense mechanisms, aimed at counteraction to such attacks carried out by potential intruder [13] is of high importance.

P. Koch et al. [5], A.J. Rae and L.P. Wildman [10] single out problems of user identification, safe data storage inside the device, installed software resistance to modification, secure access to the network, secure network connection, etc. as the key problems in the field of embedded devices security. At that, modern mechanisms of embedded systems security are oriented mainly at provision of defense against definite threats. Thus, A.J. Rae and L.P. Wildman [10], D.G. Abraham et al. [1], O. Kommerling and M. G. Kuhn [6] propose different classifications of threats and intruders for embedded devices, proceeding from intruder's abilities, his/her competence, type of access to the device; they also uncover some methods of threat prevention.

In order to prevent revealed threats in the embedded device design process functional protection properties should be provided and can be covered by some specific protection mechanisms in the form of security building blocks. However, combination of particular blocks on the basis of solely functional protection properties turns out to be ineffective because of significant resources limitations imposed by the device and hence it leads to impossibility of obtained solutions deployment in practice.

The suggested configuration model is targeted on forming the integrated protection for individual embedded devices. However, generally embedded devices function within the bounds of some macro system and the functionality of the device and related threats depend directly on the system, therefore, configuring should be regarded as a part of design process for the whole system. Besides, the choice of blocks for a group of devices interacting with each other should be conducted in coordination.

Specificity of systems with embedded devices and the configuration problem solved is at least twofold.

Firstly, we should underline the limited nature of the device resources and resulting complexity in using traditional means applied conventionally to protect PCs and servers. At that, the limited nature of the device resources arises due to both technological and computational constraints and market demands on sufficiently cheap and secure devices. Therefore, it brings the problem of achieving a sensible trade-off between embedded device security, functionality and performance. Such trade-off would allow support for the acceptable protection of the system, having constraints on resource volumes for particular devices.

Secondly, we should take into account the specific sets of attacks that can be targeted against the system with embedded devices. In addition to constraints on resource volumes, embedded devices are often characterized by properties of mobility and possibility to function in various environments with different kinds of intruders, levels of trust and types of communications changing with time. Hence, depending on a particular scenario the device can be a subject to various classes and types of attack conducted by potential intruders [13] having specific goals, skills, resources and tools. Depending on various scenarios of the system, possible changes in threats and intruders demand taking them into account within the design process.

By a configuration we imply a set of security building blocks deployed on the device and providing implementation of one or several protection functions. In essence, configuration process represents a search for such a configuration that, firstly, covers all demanded functional protection properties; secondly, satisfies the constraints imposed on volumes of device resources being allocated for protection functions fulfillment; and, thirdly, is optimal. A configuration meeting the first two conditions is called an admissible one. The optimality is meant in accordance with some optimality criterion set in the configuration process. As well, every configuration, both admissible and optimal, has to satisfy platform compatibility constraints, such as type and version of the mobile operating system of the device.

We took MARTE [14] as the basis for embedded devices analysis. MARTE comprises specification series and ontological representation of embedded devices and systems with the use of UML. In particular, in the present work we use methodological base for definition and evaluation of software-hardware resources of embedded devices and non-functional properties.

The rest of the paper is organized as follows. Section 2 of the paper is devoted to representation of the configuration model structure. Section 3 exposes main principles of embedded devices configuring on the basis of solving optimization problem. It also considers the issues of non-functional properties' modeling, configuration model application and the analysis of incompatibility between security building blocks. Section 4 is devoted to a case study. We consider the case study simulation peculiarities and experiments on evaluation of the device resources' consumption. The possibilities of the configuration software prototype are demonstrated.

2 Core Conceptions of the Configuration Model

S. Ravi et al. [12] single out a number of embedded system design challenges, such as processing Gap, battery Gap, tamper resistance, cost and others. As a rule, each challenge is tackled individually with the help of specific approaches and solutions. However, due to significant resource constraints of the devices, the design solutions should be considered together in order to minimize total expenses on security functions [2]. In other words, during evolution of the device, its functionality and security functions, the chosen set of such design solutions should be reconsidered in order to increase the performance in new conditions.

As a result, a new design challenge occurs to achieve a trade-off between "high-security and high-performance" [2]. G. Gogniat et al. [2] suggest using reconfigurable security primitives on the basis of reconfigurable data path in order to dynamically adapt its architecture depending on the system and environment states as one of the ways to achieve solution to this challenge. In fact, G. Gogniat et al. [2] suggest, firstly, to switch from one protection mechanism to another, depending on runtime requirements, and, secondly, to upgrade dynamically the protection mechanisms of the devices.

As opposed to [2], in this paper we suggest a design-time approach where obtaining of the productively efficient solutions is based on choosing the design solutions (namely, security building blocks), proceeding from non-functional resource requirements to the device and optimality criteria to overcome "high-security and high-performance" challenge.

The proposed configuration model for embedded devices is grounded on the component based approach to modeling and design, where protection is composed of a collection of single software and software-hardware modules (security building blocks), each of them being responsible for implementation of one or several security requirements [8]. The advantage of this approach is a greater flexibility of the design process, in particular, its directivity to requirements changes introduced at different stages of the design process and resulting in revisions of some stages already conducted. Another advantage of the component based approach is also the possibility to carry out conflicts and inconsistencies analysis between particular security building blocks.

As an example of practical application of the component based approach we chose mobile operating system Google Android, which implies the process of application development on the basis of reusable components, each of them being responsible for execution of certain functions.

The modeling is conducted in terms of properties of the device and security building blocks. In particular, security requirements to the device are formulated with the help of functional protection properties, while device resources demanded by security building blocks are set on the basis of non-functional properties.

Generally, the configuration process includes the analysis of properties and available security building blocks as well as the choice of those ones that will implement declared requirements in the effective, optimal way. The effectiveness is evaluated in terms of some definite optimality criterion.

In essence, the configuration process assumes interaction between the target device and the configuration consisting of a set of security building blocks (SBBs). The blocks provide functional protection properties to the device, if one allocates needed resources (Fig. 1).

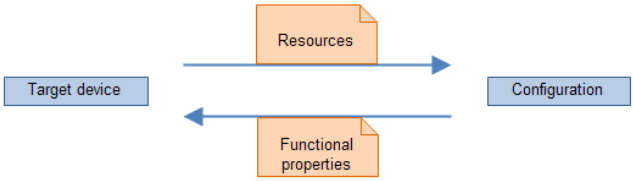

Fig. 1. Interaction between a configuration and a device

At the conceptual level the following diagram exposes the core elements of the configuration model (Fig. 2).

The diagram presents a stack based structure, where actions (or data) conducted (or formed) in the design process are drawn up from below to the top. At the bottom level the elements are responsible for getting the input data for the configuration process. At the middle level there are the ones respond to extraction of relevant input data from the specifications of the device and security building blocks. At the top leve there are the elements presenting the core functions involved in the configuration process on the basis of the lower levels.

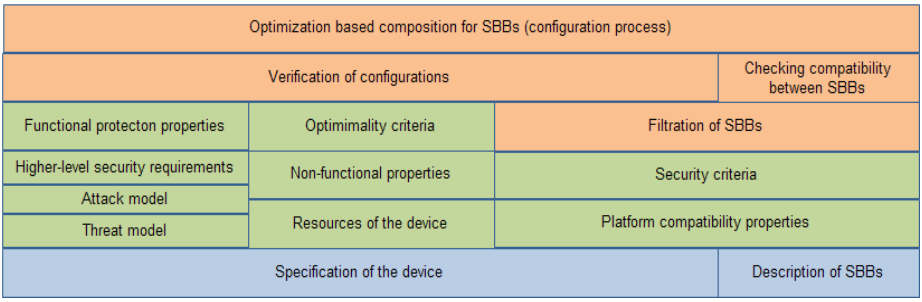

Fig. 2. Structure of the configuration model

Taking into account specification of the device and peculiarities of its environment the developer of the device forms threat model, describing goals of a potential intruder trying to compromise the device and/or the services it provides. An attack model is constructed on the basis of the threat model; for each threat it describes possible attacks targeted on achievement of the goal, including description of actions of the attack and resources needed for its fulfillment. Assaulted on counteraction of the revealed attacks, the protection is formulated by means of security requirements. Each requirement determines the necessity of carrying out one or several functional protection properties. The examples for such properties are as follows: “integrity of data stored on the device”, “authenticity of communication channel used for connections with other devices”, “implementation of remote attestation for the platform of the device”, etc.

In accordance with MARTE [14], a number of types of resources for embedded devices are regarded, particularly computing, storage and communication resources. Each resource is characterized by one or several metrics, numerical non-functional properties, which makes it possible to get quantitative values for resource consumption by each security building blocks running inside the device. The examples of non-functional properties are “volume of device’s memory allocated for execution of a security building block” (it could be measured in Mb) and “maximum speed of the communication channel” (measured in Mb/sec). Non-functional requirements are formed on the basis of worst-case resource consumption values (WCRC values). For a specific non-functional property WCRC value is assigned as the worst-case consumption value for the block.

WCRC values could be obtained experimentally on the basis of software based modeling (simulation), in particular through comparison of two modes of the application: unprotected application mode and one of applications protected by means of the given security building block. Non-functional property values for constraints of a device are identified reasoning from the specification of the device and its particular characteristics assigned by the manufacturer as well as empirically through measuring free volumes of the device resources.

At the design stage of the device, the analysis of incompatibilities between security building blocks is conducted as well. In essence, it allows revealing potential contradictions between them, which can become apparent under some specific conditions during exploitation of the system.

3 Configuration Model Development

A. Configuring by Optimization Problem Solving

The core element of the paper is the configuration process on the basis of solving the optimization problem. The peculiarity of the process lies in the fact that the choice of security building blocks is carried out not only starting from the required protection functionality, but also taking into account non-functional, resource based properties of the blocks to be deployed on the device.

Optimization problem allows finding the most effective configuration among the set of all possible ones. Efficiency is regarded in terms of the most economical consumption of some specific device resource or through minimizing an integral metric dependent on a number of resources.

Following the classic understanding, the optimization problem assumes, first, an objective function, which identified optimality criterion and, second, a set of constraints.

Objective function represents a mathematical expression (criterion expression) containing variables for one or several non-functional properties. At that, the expression is minimized in the set of admissible configurations.

We consider the following optimality criteria for the configurations:

- Minimal consumption of a resource allocated for the protection functions (e.g. minimal consumption of memory used by the blocks of a configuration).
- Optimality on the basis of a "chain of properties", i.e. the previous criterion applied sequentially for a series of resources in an order according to their significance.
- Other integral criteria relying not only on the resources consumed by configurations, but also on their device constraints. For example, the following formula declares a criterion that allows a device developer to detect a configuration maximizing the minimal free amount over all the resources (calculated in percent):

$$\min_i \{ \max_{p \in NonFuncProperties} \{ percent(p(configuration_i)) \} \},$$

$$percent(p) = \frac{constr(p) - p}{constr(p)},$$

where p denotes a non-functional protection property numerically characterizing some resource of the device; *percent*() function returns a normalized value of the property in compliance with its constraint *constr*().

All the constraints on the objective function can be divided into three groups:

- Constraints on functional protection properties, binary ones determining whether a particular security building block covers some protection function or not.
- Constraints on numerical values of non-functional properties characterizing needs of the block in a specific resource formulated in the form of inequalities with the use of WCRC values (e.g. memory consumed by the configuration should not exceed a particular value).

- Compatibility constraints between a block and the device platform in the shape of so-called platform compatibility properties. These constraints determine presence or absence of any platform peculiarities, such as support for a specific type operating system and its version, embedded CPU type [11], presence of network wire and wireless interfaces like Ethernet, Bluetooth, etc.

Generally, the optimization problem solved is a multi-objective extreme task with the identified set of constraints. Its formal settlement is formulated with the use of set-theoretic presentation:

$$Opt\_criterion(non\_functional\_properties(configuration)) \rightarrow min/max$$
$$Constr(functional\_properties(configuration))$$
$$Constr(non\_functional\_properties(configuration))$$
$$Constr(platf\_compat\_properties(configuration))$$

Here by *Opt_criterion* we mean some optimization criterion being either minimized or maximized. At that it depends on the constraints (*Constr*) of the properties of configurations.

B. Modeling Non-functional Properties

Both functional protection properties and platform-compatibility properties are binary ones, and their values are defined by choosing one of two predetermined values, while non-functional properties demand more detailed consideration.

In this paper we are based on the definitions and methodological foundations proposed by MARTE [14]. Thus, in accordance with MARTE, the measurable quantitative non-functional properties are singled out.

At that, firstly, quantitative property is characterized by a set of Sample Realizations that are determined (measured) for a property during runtime, while the experimental measurements can be conducted on the real system or on the basis of software modeling (simulation). In particular, for cyclic deterministic systems such values might be obtained singly and “extrapolated” to the subsequent time cycles.

Secondly, a non-functional property is characterized by so-called Measure function allowing juxtapose some numerical value to a set of gained values. Measure function could be, for example, some mathematical function like max (maximization of set), min (minimization), mean (averaging function).

For example, the procedure of getting the values of the non-functional property “volume of consumed memory of the device” of some security building block might be carried out the following way:

- Points in the software (time-points) to take measurements are chosen.
- Block’s action is initiated, measurements are taken.
- Maximization function producing required value is applied.

To obtain values of non-functional properties the security building blocks should run in debugging mode with the use of profilers or measuring procedures could be built into the application. In the last case it is reasonable to take into consideration the side-effects of these procedures and possibly correct the obtained values.

In accordance with MARTE, the following types of hardware resources and non-functional properties corresponding to them are singled out:

1. Computing resource (*HW_Computing package*), its main characteristic being opFrequencies property which defines interval for values of CPU frequency, which it can function on [14]. The resource is also characterized by MIPS, FLOPS values allowing estimate the number of operations that can be conducted in a specific time gap.
2. Memory resource (*HW_ProcessingMemory*) which is characterized, in particular, by the memory volume and the response time [14].
3. Storage resource (*HW_StorageManager*) which is characterized by the storage volume.
4. Communication resource (*HW_Communication*) is characterized by the channel bandwidth.
5. Energy resource (*HP_Power*), being spent in particular on the security building blocks activity and on the heat dissipation. An energy resource is characterized, firstly, by the capacity of the power source necessary for the device activity (*HW_PowerSupply*), and, secondly, by the capacity of the battery-driven resource which determines the duration of the autonomous work of the device (*HW_Battery*) [14].

Beside resource properties, the so-called "external" properties stipulated by some external requirements to the device functionality (such as, for example, blocks cost, their physical characteristics like weight, size, etc.) could also be referred to non-functional properties and serve as the basis for configuring.

C. *Configuration Model Application*

The configuration scenario is oriented at the embedded device developers and encompasses search for the optimal configuration under some invariable constraints on the volumes of device resources (Fig. 3).

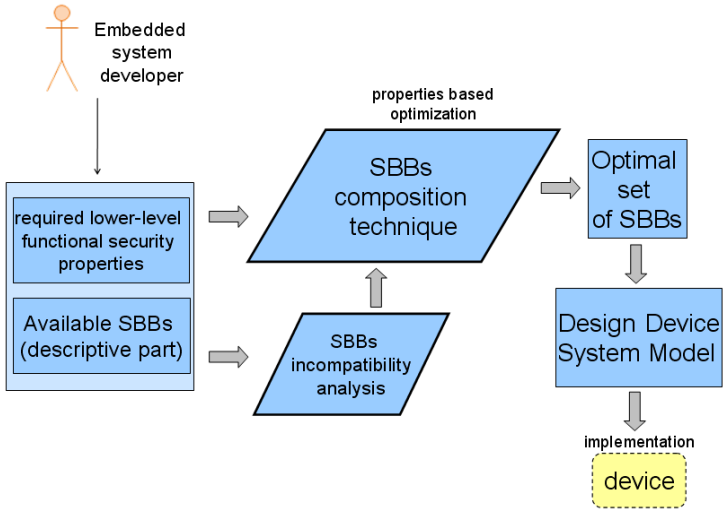

Fig. 3. Configuration model application scenario

The scenario is formulated as “a choice of the most effective configurations on the basis of properties of security building blocks and constraints of the device”.

Table 1 shows information on the configuration scenario, including roles, stages, activities and conditions.

Table 1. Configuration scenario

Scenario	A choice of the optimal configuration(s)
Roles involved	Embedded device developer
Activity	Design and deduction of needed security building blocks
Conditions	♦ Great number of basic functional protection properties to be covered ♦ Huge amount of alternatives in choosing blocks implementations ♦ Given non-functional (resource) limitations of the device, which cannot be violated ♦ Changing functional protection properties (including adding new ones) during Engineering Process

D. Analysis of Incompatibilities Between Blocks

Beside requirements to the rest part of the device a security building block can make demands to other blocks as well. Thus, individually each block can give necessary functionality, however requirements between particular blocks are not met. As a whole, the paired incompatibility between blocks can be determined by means of a square matrix (Table 2).

Here pairs {SBB1, SBB2} and {SBB1, SBB3} are compatible, whereas {SBB2, SBB3} are incompatible. Generally, such matrix is prepared by security experts and used by developers of the device to reveal conflicts appeared between the blocks. By “blocks” here we mean not only SBBs granting some specific security functions to the device, but also “engineering” (non-security oriented) blocks presenting business functions such as data storage modules, communication blocks, etc. The main task is to discover incompatibilities in the design process of the device.

Table 2. Table of compatibilities between security building blocks

Block compatibilities			
Compatibility	***Block 1***	***Block 2***	***Block 3***
Block 1	+	+	+
Block 2	+	+	–
Block 3	+	–	+

Fig. 4 depicts two types of incompatibility described below.

Generally, the design engineering process includes the following actions:

1. Specification on the basis of the expected business logic of the device, its specification is carried out.
2. Configuring checking that each block is compatible with the device is conducted on the stage of SBBs integration into the device. First, it should be

verified that the device is capable to provide chosen blocks with all necessity resources. Second, it should be checked that each block is compatible with software/hardware platform of the device (operating system and its version, sockets, etc. should be conformed).

Particularly, type 1 inconsistencies can arise due to the insufficient coordination between business logic and specification of the device.

Type 2 inconsistencies represent the contradictions between protection functions belonging to different blocks.

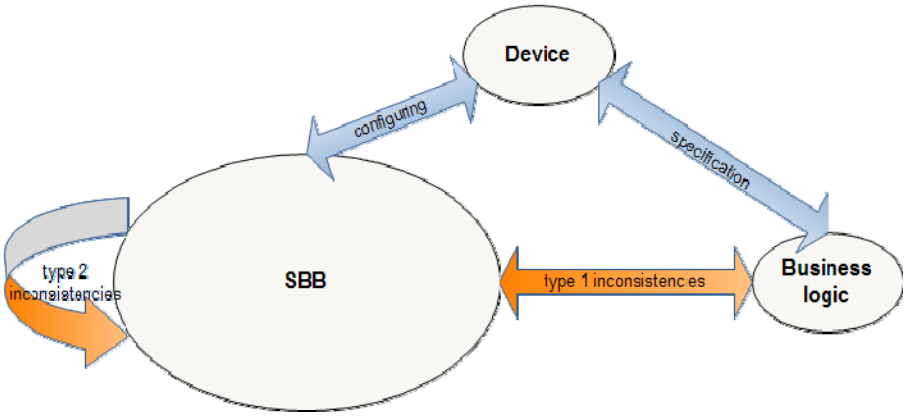

Fig. 4. Types of Incompatibility

Let us consider these incompatibility types:

1. Incompatibilities between the block and business functions of the device.

- Example. Let us regard a functional protection property "backup of customer data stored on the device". Assume in compliance with business requirements to the device the volume of data stored on the device can reach 1 terabyte. However, the corresponding backup-block available in the device allows storing lesser amount of data only. Thus, such kind of incompatibility is a conflict between a SBB and the business functions of the device being represented by some other "system" blocks.

2. Inconsistencies between functional protection properties of several SBBs. Here we regard logical incompatibility between SBBs, i.e. a contradiction between functionality of SBBs.

- Example. Assume there are two SBBs, a backup-block (block 1) and a block implementing secure guaranteed deletion of customer data after some specific event happens (block 2). Inconsistency lays in the fact that block 2 works appropriately only with special interaction with block 1. As a result such inconsistency can be eliminated through conducting a specific scenario of SBBs integration and checking correctness of their combined usage within the device.

We should be able to detect "potential conflicts" as well, i.e. such inconsistencies between blocks that became apparent merely under some specific conditions.

4 Case Study

The goal of the case study we provide is to demonstrate applicability of the configuration principles in practice. The case study represents, firstly, a demonstration example and, secondly, a software prototype of the configuration mechanism. The demonstration example contains software based simulation of some application running on a device and its environment. Particularly, the example assumes carrying out techniques that allow obtaining the values of non-functional properties for blocks and the resource constraints of the device. In essence, the role of the case study is to provide the configuration model with proof-of-the-concept and show a simplified example that could be used by device developers as a pattern.

A device used in simulation process is a smart phone HTC Wildfire S on the basis of Android 3.2 platform. The application running on the device is some sort of network "messenger" communicating on Internet. The security goal is to protect it against unauthorized modifications (Fig. 5).

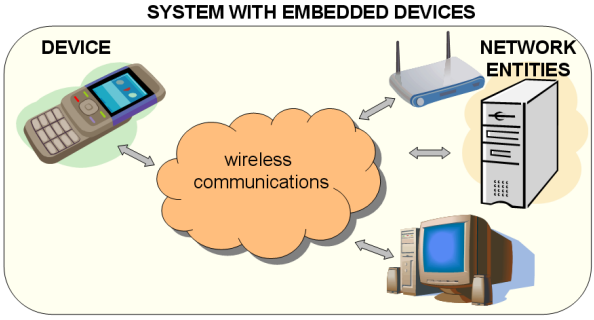

Fig. 5. Demonstration Example

To simplify the experiment we are considering merely two functional protection properties. The first one is formulated in terms of necessity of implementing a function of continuous monitoring of the internal state of the application by means of remote attestation [4]. The second property is confidentiality of critical business data stored in the read-only memory.

A. Simulation

Simulation comprises implementation of the application, security building blocks integrated into it and measuring functions for non-functional properties. Conducted simulation and experiments have the following limitations and assumptions.

Because of weak maturity of profiling tools for Android platform non-functional property measuring functions are built into the application and considered as a business functions. We assume that execution of these functions does not influence significantly the measurement results, i.e. we can neglect resource expenses on them. Evaluation of the expenses of a device resource allocated for the support of a block is carried out through measurement and comparison of non-functional properties for two variants of the application, an unprotected program and one protected by the block. In

the first case we determine resource expenses on the business functions, whereas the second one allows us to get summarized expenses on both business functions and security functions.

We are taking the following assumption as well. Since values of non-functional properties are measured individually for each block, we consider the summarized values for configurations are sums of the corresponding values for every block which is a part of configuration. Meanwhile, in practice it is reasonable to consider such mutual influences between blocks. For example, remote attestation block could check not only business functions, but other blocks as well.

In simulation we are limited by three types of resources of the device as the simplest ones for technical implementation and experiments: memory resource, storage resource, and communication resource. Consequently, three non-functional properties are regarded:

- Maximal amount of memory used by a block during its functioning.
- Minimal volume of read-only memory required for storing the block and supporting its execution.
- Minimal bandwidth capacity of communication channel demanded by the block.

Analogously three non-functional constraints of the device are considered:

- The size of the device memory allocated for the blocks.
- Volume of the device storage allocated for the block support.
- Bandwidth the communication channel is able to provide to support functioning of the blocks.

Table 3 summarizes data on techniques used in the experiments in order to compute non-functional property values for each of resources.

Table 3. Analysis of non-functional properties

Resources	**Non-functional properties**	**Techniques for gathering/measuring values of non-functional properties**	
		Values for blocks	***Constraints of device***
Memory resource	Volumes of available / required memory in MB	Gathering / measuring data in runtime by means of measuring functions built in the application	Deducing through analysis of device specification
Storage resource	Volumes of available / required storage resource in MB	Immediate read of file size attribute	Through analysis of device specification
Commu-nication resource	Volume of available / required Network connection bandwidth	Analytically through analysis of block specification with the object to send / receive data within established time gap	Through analysis of device specification

For storage resource computation we regarded only the size of a software module presenting the block. At that we did not took into account such costs as expenses on memory virtualization (swapping) and ones on storing temporary or persistent security relevant data connected to the block.

For the communication resource we assumed only synchronous communication. At that, in the experiments we use the timeliness value, which is reasoned from the internal security requirements of the remote attestation block and assigns a check time gap for the data to be transmitted.

B. Results of Experiments

Conducted experiments assume modeling the work of two types of blocks supplied with procedures of resource consumption measurement.

In conformity with the experiment conditions, the modeled application contains a number of data structures regarded as critical in the movement of unauthorized modifications. To make it simpler in technical implementation we consider data structures of the same size, namely 400 instances of byte arrays, each one of 100 bytes. For the remote attestation block we suppose each data structure assumes several admissible values regarded as correct ones. If some data holds a different value, it means there is a violation in the application.

For each data structure a hash value (by means of MD5, SHA, etc.) is computed, sent to the attesting entity side and subjected to verification. We assigned the check time gap parameter equaled to 1 second (i.e. every second a set of signatures in the form of hash values should be transmitted to the remote attesting entity).

According to the hash function bitness and applying math multiplication operation we obtain that with the help of MD5 the volume of passed data makes up 400 * 128 = 51200 bit/sec, while for SHA-256 we get 102400 bit/sec (here we neglect overhead data). For the resource constraints we suppose that data are transmitted with the use of GPRS channel, and hence we let the minimal ensured speed of the channel equals to 32 Kbit/sec.

The consumption values obtained for each block are expressed in Table 4 and Table 5.

Table 4. Resources consumed by remote attestation block

Remote attestation	Memory resource consumption (KB)	Storage resource consumption (KB)	Communication resource consumption (Kb/sec)
with the use of MD5 hash 128 bit	287	4	51,2
with the use of SHA-256 hash 256 bit	359	4	102,40

Table 5. Resources consumed by symmetric encryption block

Symmetric encryption of critical data	Memory resource consumption (KB)	Storage resource consumption (KB)	Communication resource consumption (Kb/sec)
AES/128 based encryption	523	4	–
DES/56 based encryption	552	4	–

Experimentally obtained values for non-functional resource related properties allow providing each of the blocks available at the device developer with the series of its resource consumption characteristics. As a result, choosing the appropriate optimality criterion, the developer gains optimal combination of security building blocks (these configurations appear to be optimal by construction).

At the great number of functional protection requirements and the number of blocks available, procedures of developer's manual complete enumeration of all possible blocks turns out inefficient, in particular, due to its exponential complexity. Carrying out automated configuration process with the use of the developed configuration tool allows finding optimal configurations in the acceptable time period. Thus experiments showed, for example, the configuration process with more than 100 variants blocks on the ordinary PC takes less than 1 second. Moreover, even with the number of blocks increasing to 1000 and more (assumed theoretically) time expenses turn out to be quite acceptable with configuring being anyway an element of the design time process.

C. Implementation of the Configuration Model

The developed software prototype enables us to demonstrate the proposed configuration approach and represents a software tool to configure distributed systems with embedded devices. The objective of the tool is to facilitate the developers in taking security-aware decisions at the design stage to provide necessary security and admissible consumption of device hardware resources. The proper configuration function allows for a given set of security blocks to reveal the optimal configuration under specific constraints and optimality criterion. The tool enables the developers to check optimality and admissibility of a configuration as well as determine what kind of the resources is the most critical for a configuration and deduce non-functional limitations for the device to apply a given configuration.

Fig. 6 depicts a fragment of graphical user interface of the configuration tool implemented. In particular it exposes the main window providing the developer with information on functional and non-functional properties of security building blocks, information on platform compatibility properties, available optimality criteria, specification of the device and some means for the governance of the configuration process.

The configuration tool application represents the final step at the security design stage for the device. Starting from the phase of security and non-security requirements, producing the simulation and carrying out the experiments allow the developer to accumulate needed data on particular blocks, configurations and the device. Afterwards, as input data for the tool these ones are used to obtain the sought solutions.

5 Conclusion

Conventionally in practice embedded device security issues are either neglected or considered only at the final stage of the design process in the form of integration of some particular security functions like encryption of the outgoing traffic into the device.

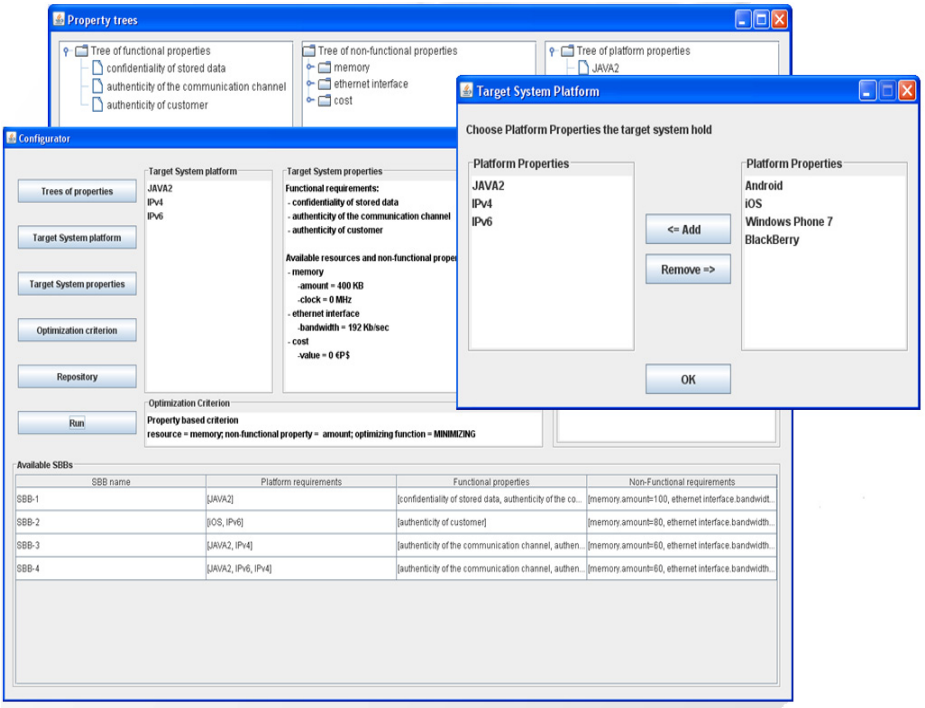

Fig. 6. Configuration Tool GUI

In the paper we proposed a new configuration model, which facilitates the design of secure and resource consumption efficient embedded devices. The proposed configuration model gives the developers a possibility to construct more secure and energy efficient embedded solutions. The model enables the search for the most resource efficient combinations of security building blocks on the basis of solving optimization problem.

We analyzed the case study simulation peculiarities and experiments on evaluation of the device resources' consumption. The possibilities of the configuration software prototype implemented were demonstrated.

The distinctive peculiarity of the proposed approach is construction of the combined protection based on particular security building blocks, taking into account both functional and non-functional protection properties. Considering non-functional ones allows device developers to deduce the needed protection, reasoning from amounts of resources available for the blocks.

The future work will comprise reinforcement of modeling and experimental part through the expansion of security building blocks under investigation. Besides remote attestation and symmetric encryption blocks, we assume to model deeply and make use of some other ones and wider range of crypto primitives they are based on. Having more security building blocks will allow us, firstly, to enhance the proof of the whole configuration concept and, secondly, to reproduce completely the configuration process proposed on the basis of the existing demonstration example.

Acknowledgements. This research is being supported by grant of the Russian Foundation of Basic Research (project #10-01-00826-a), Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (contract #2.2), State contract #11.519.11.4008 and partly funded by the EU as part of the SecFutur and MASSIF projects.

References

1. Abraham, D.G., Dolan, G.M., Double, G.P., Stevens, J.V.: Transaction security system. IBM Systems Journal 30(2), 206–228 (1991)
2. Gogniat, G., Wolf, T., Burleson, W.: Reconfigurable Security Primitive for Embedded Systems. In: Proceedings of International Symposium on In System-on-Chip, pp. 23–28 (2005)
3. Grand, J.: Practical Secure Hardware Design for Embedded Systems. In: Proceedings of the 2004 Embedded Systems Conference, San Francisco, California (2004)
4. Gu, L., Ding, X., Deng, R.H., Xie, B., Mei, H.: Remote attestation on program execution. In: Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing (STC 2008). ACM, New York (2008)
5. Kocher, P., Lee, R., Mcgraw, G., Ravi, S.: Security as a new dimension in embedded system design. In: Proceedings of the 41st Design Automation Conference (DAC 2004), San Diego, CA (2004)
6. Kommerling, O., Kuhn, M.G.: Design principles for tamper-resistant smartcard processors. In: Proceedings of the USENIX Workshop on Smartcard Technology, Chicago, pp. 9–20 (1999)
7. Koopman, P.: Embedded System Security. IEEE Computer (2004)
8. Kuntze, N., Rudolph, C.: Secure Digital Chains of Evidence. In: Proceedings of 2011 IEEE Sixth International Workshop on Systematic Approaches to Digital Forensic Engineering, SADFE 2011, Oakland, CA, USA (2011)
9. Lee, G.M., Kim, J.Y.: The Internet of Things – A problem statement. In: 2010 International Conference on Information and Communication Technology Convergence (ICTC), pp. 517–518 (2010)
10. Rae, A.J., Wildman, L.P.: A Taxonomy of Attacks on Secure Devices. In: Australian Information Warfare and IT Security, Australia, pp. 251–264 (2003)
11. Raghunathan, A., Ravi, S., Hattangady, S., Quisquater, J.: Securing Mobile Appliances: New Challenges for the System Designer. In: Proceedings of DATE 2003, pp. 3–7 (2003)
12. Ravi, S., Raghunathan, A., Kocher, P., Hattangady, S.: Security in Embedded Systems: Design Challenges. ACM Transactions on Embedded Computing Systems 3(3), 461–491 (2004)
13. Ruiz, J.F., Harjani, R., Maña, A., Desnitsky, V., Kotenko, I., Chechulin, A.: A Methodology for the Analysis and Modeling of Security Threats and Attacks for Systems of Embedded Components. In: The 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2012), Munich, Germany (2012)
14. Object Management Group: The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems. Version 1.1 (2011), `http://www.omgmarte.org`

A Study of Entropy Sources in Cloud Computers: Random Number Generation on Cloud Hosts

Brendan Kerrigan and Yu Chen

Dept. of Electrical and Computer Engineering, SUNY - Binghamton

Abstract. Cloud computing hosts require a good source of cryptographically strong random numbers. Most of the standard security practices are based on assumptions that hold true for physical machines, but don't translate immediately into the domain of virtualized machines. It is imperative to reconsider the well accepted security practices that were built around physical machines, and whether blind application of such practices results in the possibility of a data breach, machine control, or other vulnerabilities. Because of Cloud computers reliance on virtualization, access to the hardware based random number generator is restricted, and virtualization can have unforeseen effects on the operating system based random number generator. In this paper, the entropy pool poisoning attack is introduced and studied and a Cloud Entropy Management System is proposed. Extensive experimental study verified that there are measurable problems with entropy in Cloud instances, and the management system effectively solves them.

1 Introduction

A rapidly growing trend is the offloading of computing resources from internally owned and operated infrastructure to the Cloud. Outsourced computing is often cheaper and incurs less business overhead than maintaining private computing infrastructure. Businesses may also build private Clouds, which often reduce the amount of infrastructure necessary by increasing the utilization of computing resources. Cloud computing also brings new challenges, and the aggressive adoption of Clouds could lead to a large population of highly concentrated machines that are vulnerable to different attack vectors.

The combination of multiple users on a single physical machine raises many security concerns. Many of these issues are related to issues that operating systems had to address when they went from single user to multiple user systems. Of main concern is keeping one user's data, activity, programs, and resources separate from all others. In the Cloud arena, this means keeping virtual machine information and state separate from other virtual machines, and insofar as it's possible to keep the virtual machine activity and state separate from the controller that schedules operating systems.

Creating such secure partitions while maintaining the advantages and flexibility of Cloud services is a difficult task. Other issues include the migration of

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 286–298, 2012.

data across networks (sometimes instances are moved due to heavy local machine load), inheritance of security vulnerabilities from platforms used to construct Clouds, trust of the Cloud operators (an unavoidable concession in public Clouds), and new vulnerabilities that emerge from a Cloud's architecture and implementation. Creation of fully homomorphic encryption systems is an essential key tosolving the trust problem of public Cloud operators. Such a system would allow users to encrypt their data, send it to the Cloud for some computational purpose, and the Cloud would perform that computation on the encrypted message, returning an encrypted result. This protects the data because it is never decrypted on the shared system. Inherently the robustness of such a system relies on truly physical random sources. It is critical to identify, demonstrate, and provide a solution to securing the possible vulnerabilities.

The main focus of this work is to explore evidence of weaknesses in the generation of random numbers in Cloud hosts, and provide tools for mitigation of these weaknesses. A thorough background on the generation of random numbers is presented, along with the reasoning why currently accepted methods for random number generation don't translate immediately into the Cloud environment. A series of experiments has been conducted to reveal statistical weaknesses in the way random samples are processed by Cloud guests and the controlling operating system. To address these weaknesses, a customized Cloud Entropy Management System is designed and implemented.

2 Random Number Generation

This section provides a brief explanation on the generation of random numbers on computer systems, focusing on the difference between a true random number generator (TRNG) and a psuedo-random number generator (PRNG) including the PRNG used in the Linux kernel.

Currently computers generally rely on two sources to gather randomness, one which measures hardware noise (hardware RNG), and another that conditions user-input to derive a certain amount of randomness which is used to seed a pseudo-random number generator. An interesting contrast with most computer functionality, generation of random numbers is actually very slow.

Corresponding to the two sources, there are two commonly recognized types of random number generators in computers. The first, a TRNG, measures some physically random features to generate bits. These are bits of entropy; true randomness, especially following some conditioning for the elimination of biases. The second, a PRNG, is a deterministic algorithm which produces streams of random appearing numbers based on some random input 'seed.'

True Random Number Generators (TRNG). TRNGs are somewhat of a burden to semiconductor designers because the standard design requires analog components, at a large power consumption price, and also considerable redesign costs because the analog designs don't scale down quite as neatly as the digital counterparts when a new process technology is transitioned to (e.g. 45nm to 28nm) [1].

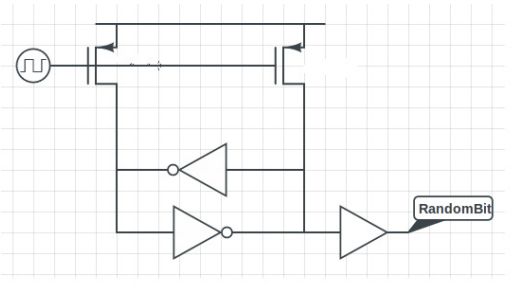

Fig. 1. Schematic of random source used in Intel's Bull Mountain Digital Hardware RNG

The basic theory of operation behind a TRNG is amplifying some noise in the system, generally the thermal noise in a resistor, and sampling that signal. This very basic design has some shortcomings such the rate of bit generation and power consumption, and Intel made improvements to it [1]. Instead of simply sampling the noise, the amplified noise was used to control a voltage controlled oscillator. Another oscillator, at 100x the frequency, is then used to create a two bit signal. To remove possible biases, a "von Neumann corrector" is used. The output of the corrector is then run through a secure hash algorithm before it is accessible from its application interface. It could produce random bits at 300Kbit/sec.

Intel has again looked to create a better random number generator recently, this time side stepping the problems with TRNGs to the present by creating an all digital generating circuit [2]. This alleviates the power consumption of analog amplifiers, and simplifies the moving of a design between fabrication technology. Their solution, code named Bull Mountain, violates common digital design practices used to ensure stability in the name of creating randomness. The circuit used as a random source is shown in Figure 1. Note that each input node of the inverter has two drivers, the output of the other inverter, and the drain of a pFET with a clocked gate. When the clock is low, both inverters have their input forced to high. This is where the circuit is in a metastable state, and the final stable output will be detemined by thermal noise that exists because of the resistive losses in the transistors. Each time the inputs are forced into that metastable state, one bit of randomness is produced. Again these raw random measurements are not used directly, but go through some conditioning like the von Neumann corrector. Once finished conditioning, the output should then be used to seed a quality PRNG.

Pseudo-Random Number Generators (PRNG). Psuedo-random number generation is the use of deterministic algorithms to produce a seemingly random sequence of numbers. By itself, a PRNG is a poor source of randomness, as knowledge of the seed value is all that is needed to reproduce the output. This

makes the secrecy of the seed paramount. Other attacks are also described that PRNG algorithms can be vulnerable to [6].

There are also a number of different PRNG algorithms, and parameters for the algorithms can have an enormous effect on the statistical quality of the output. Being the goal is a uniform distribution, a good PRNG will generate each number in the entire range, regardless of seed. A chronological background of selected popular generators follows.

For many years, PRNGs were created based on Prime Modulus Multiplicative Linear Congruential Generators (PMMLGC) [13]. A generator, g, is an element of a group, G, which when raised to integer powers, generates a cyclic subgroup belonging to G [15]. In this case, the group is the integers (without 0) over a prime modulus. This relies on some number and group theory results.

A very efficient PRNG can be constructed from shift registers with feedback. These generators rely on bitwise shifts, and usually the XOR operation, and are in general called General Feed-back Shift Registers (GFSRs). However, they do perform poorly on some statistical tests for randomness, such as initialization sensitivity and partitioning problems [3].

An interesting solution to the problem of apparent parallel hyperplanes in these generators is to carefully combine the output of two generators, creating what is a called a combined PRNG [14]. The resultant sequence of pseudo-random numbers, after combination, doesn't exhibit the apparent parallel biasing.

Two less traditional approaches have been proposed in [4] and [5]. In [4] a cryptographically secure pseudo-random number generator, named Yarrow, is created. The main idea in Yarrow is to keep a more conservative estimate of the real entropy gathered from the system, and only keep an entropy accumulator, as opposed to a giant pool with the samples mixed in. The accumulator is then used in combination with one-way hash function to provide random numbers. The width of the accumulator is a major drawback to high performance applications.

The designers of Yarrow relooked at the problem a few years later, with a focus on removing need for highly precise entropy estimation. Their new scheme, named Fortuna [5], relies on keeping 32 pools of entropy, and spreading the entropy across the pools in an even manner.

The Linux RNG (Twisted GFSRs). The Linux kernel provides two character devices which output random numbers, /dev/random (blocking) and /dev/urandom (non-blocking). Cryptographic services are recommended to avoid the use of the non-blocking random numbers, as they could be open to state compromise extension attacks [6]. The quality of the outputs of the random number generator relies heavily on the entropy provided by the input.

The Linux RNG collects entropy by measuring inter-interrupt timing from various sources, mainly the keyboard, mouse, disk read and writes, and network interrupts [7,8,9]. Cloud instances are starved of the first three, and are largely dependent upon network interrupts for entropy. This has significant ramifications for the security, because these interrupts not only contribute to the virtual machines entropy pool, but also to the scheduling operating system's entropy pool.

The inter-interrupt timings are used as input, along with the current pool contents, to a twisted General Feedback Shift Register. The output is fed back into the pool. Finally if a read of the pool is requested, the pool is used as the input to the Secure Hash Algorithm (SHA). This step is to provide further security to the secret state of the entropy pool, as SHA is considered to be irreversible. While the SHA is currently considered good in that respect, there is a possibility of the algorithm being analyzed where attack is possible. It would be desirable to run SHA on the entropy samples. However, it would have serious overhead when being called so frequently.

3 Distributed vs. Shared Entropy Distribution

In this section, scheduling in the Xen hypervisor is discussed to provide the necessary understanding of how virtualization scheduling leads to hypothetical weaknesses in random number generation in virtual machines, and the virtual machine monitor itself. When it comes to entropy sources on cloud systems, each level of service may offer access to a different type of source. For Infrastructure as a Service (IaaS) systems, entropy sources are distributed among the instances; each instance generates its own random numbers. For Software as a Service (SaaS) and Platform as a service (PaaS), it is possible that all services share a common pool of random numbers.

There are advantages and weaknesses to both approaches. The advantage of a distributed entropy generation is the control in selecting how random numbers are generated, and the separation from possible adversaries. It does however open up the possibility of attacks on the privileged operating system that schedules instances. In shared entropy systems, it is possible for a malicious user to drain entropy from the entropy pool faster than it can be filled, leading to performance degradation of other users and possible denial of service. If a shared entropy system is unable to create random numbers at a rapid enough rate, it would stand to limit access to the pool, or even market the access. In cloud computers that leverage the Xen virtualization hypervisor, high entropy sources are scarce. This scarcity will be explained in the next section.

3.1 Xen Scheduling

The Xen system has a multitude of different schedulers, which use various mechanisms to decide which operating system gets scheduled next. The scheduling of operating systems is analogous to the scheduling of processes within an operating system. The main three schedulers that Xen uses are the Borrowed-Virtual-Time (BVT) scheduler, the Earliest Deadline First Scheduler (sEDF), and the Credit scheduler [12]. There is also a round-robin scheduler, which will be used to simplify the presentation of attacks. The BVT scheduler allows latency sensitive applications to jump the scheduling queue, at the cost of owing that CPU time at a later point [10].

In the sEDF scheduler, anytime a process is scheduled, the priority queue of processes is searched for the one with the most imminent deadline [11]. Both of

these schedulers are to be deprecated in future Xen versions. The credit scheduler is the most recommended scheduler by the Xen maintainers, and tasks are scheduled based on a currency based system where Virtualized CPUs (VCPUs) are scheduled in a queue, and 'pay' for real CPU time. Each VCPU receives an allowance from an accounting system.

In Section 6, we use a round robin approximation, assuming an attacker saves his VCPU credits (used as a score in normal Xen scheduler) for a few rounds of scheduling, and hence has a reliably long window to execute the attack.

4 Attacks on Cloud Entropy Sources

Two possible attacks on cloud entropy were conceptualized. One is the random number pool depletion attack that focuses on a shared entropy distribution architecture like that of a PaaS cloud might have, and another one is the entropy pool poisoning attack that focuses on a distributed entropy distribution architecture like an IaaS cloud would likely have. The latter is the focus of this paper.

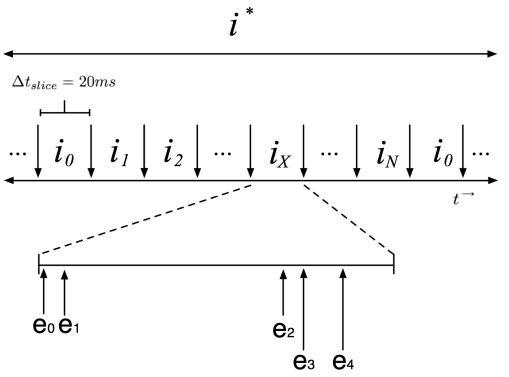

Fig. 2. Overview of Scheduled Pool Poisoning Attack

Pool Poisoning Attack. The entropy pool poisoning attack is a theoretical attack with dire consequences for victims, from attackers being able to decrypt secured traffic (exposing for instance, credit card numbers) up to arbitrary code execution through remote procedure calls (RPCs). From a high level, the attacker tries to make contributions to another user's entropy pool, by generating interrupts with known delays between them. Those delays are run through the Linux RNG as input, and provide some knowledge of another user's entropy pool contents.

$$
\begin{aligned}
I &= \{i_0, i_1, i_2, \ldots, i_x, i^*, \ldots, i_{n-1}, i_n\} \\
E &= \{e_0, e_1, e_2, e_3, e_4\} \\
\tau &= (\Delta t_0, \Delta t_1, \Delta t_2, \ldots, \Delta t_m)
\end{aligned}
$$

Where:

- I: set of n instances on the cloud host
- i_x: attacker instance
- i^*: dom0 operating system
- E: set of attack events (enumerated in detail below)
- τ: Sequence of inter-interrupt timings generated by i_x during e_1

Note that with respect to interrupts, i^* "sees" all the interrupts that the currently scheduled instance "sees". Below is an enumeration of the events in E:

- e_0 Flush the entropy pool (deplete until empty, then stop)
- e_1 Begin sending TCP packets.
- e_2 Stop sending TCP packets.
- e_3 Read the entire entropy pool.
- e_4 Transmit or store before time slice ends.

For clarity, note that in e_1 every TCP packet that is sent is followed by the next Δt_i over the entire sequence τ. Following these steps, the entropy pool of i^* should be full of the contributions matched in the copy of the entropy pool that is stored or sent in e_4.

5 Cloud Entropy Management System

In this paper, a Cloud Entropy Management System is proposed to address the weaknesses of entropy generation on Cloud instances. The Cloud Entropy Management System was created to operate on the cloud hosts to help provide guests with more refined estimates of entropy in their pool. It also provides a mechanism for getting rescue entropy in the case where entropy production from the cloud hosts is insufficient to provide for a workload that requires a large quantity of entropy. This design uses the philosophy of least intervention, allowing the existing kernel PRNG to operate normally, but giving it a more realistic estimate of the entropy in its samples. This allows the kernel PRNG to operate under the assumptions the PRNG creators outlined in the source file (namely that secrecy of entropy contributions is maintained and accurately estimated.) The amount of true entropy bits in a physical cloud host is not very straightforward. In general estimating entropy is difficult, however, it is important to have a reasonable estimate, as this is used as a parameter in the number of samples mixed into the pool. The Linux kernel provides easy access to the estimate in the proc filesystem. The entropy management system looks to provide a more reasonable entropy estimation to overcome accumulation of errors in each sample. Figure 3 shows the operation of the system. Each cloud host runs a server which provides service to the instances that run on it. If a cloud instance on the server is running the Entropy Manager, it will connect on start up to the server. The server provides a count of the number of instances on the cloud, which is used as a divisor for the entropy estimate on the guest. This is a consequence of the principle earlier mentioned that the sum of the entropy

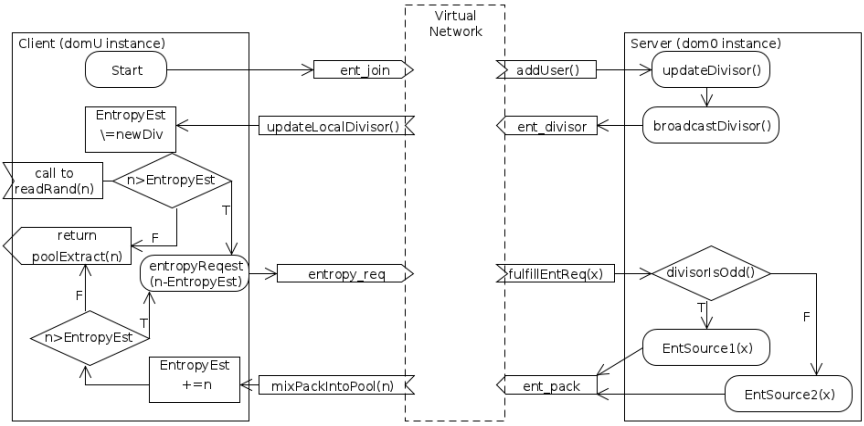

Fig. 3. Activity diagram of the Entropy Management System

of the virtual machines cannot exceed the entropy of the physical host. This divisor is broadcast to clients.

Consequently, the Linux PRNG will use this updated estimate to provide appropriate feedback to internal thresholds used to control generation. Finally, in order to provide for flexibility, the client can request entropy packages from the server. For testing, one source of entropy is a HTTP GET request to random.org for 2KB of digitally sampled atmospheric noise.

There are some security concerns (namely packet snooping between the cloud host and random.org) with this, but it is only used as a proof of concept. For instance, if an attacker was able to intercept the HTTP response from random.org, and knew the hash function used, the system would provide no extra security. There are methods that could be used to overcome this fairly easily, such as employing encryption between the two, or even adding some local entropy to the sample before the hash function.

The server program is designed to allow flexible source selection. This was done in anticipation of new sources becoming available, such as the Bull Mountain source [2]. Emergency entropy is added to the entropy estimate, then the total is checked again to ensure that the estimate hasn't changed in the mean time. Following this, a read is allowed to return.

6 Experimental Study and Results

Building an Experimental Cloud. Initial work was done on the Amazon EC2 cloud, however flexibility required for experimentation required a private Xen Cloud Platform to be setup. The test cloud host was small, however it is sufficient for the testing required. The private cloud host consisted of a Dell

PowerEdge 2950 server with the following specifications: 2x Quad-core Xeon E5335 Processors @ 2GHz, 8GB @ 667MHz, and 500GB of Storage.

6.1 Pool Poisoning Attack

The pool poisoning experiment revealed the source of entropy weakness in cloud guests. The entropy contributions used in the Linux RNG consist of the *jiffie, num, and cycle* variables. Each interrupt causes a read on these variables. *Jiffie* is a counter of the number of timer interrupts generated (every 4ms in Linux). *Num* indicates the type of interrupt, such as keyboard, or network. *Cycle* is a free-running counter register available in x86 CPUs that runs at the clock frequency.

Invariability of Jiffie and Num. In Figure 4, the invariability of both the jiffie and num variables with 4 guest instances are demonstrated. After the first 300 seconds, the jiffies quantity is invariant for both domain 0, and the guest instance. This problem is likely filtered by the hypervisor, so no guests are affected by it. The num variable however, has significant change in all cases for domain 0, but is completely static for all guest instances. The same invariability is observed in the experiment over one and two guest instances.

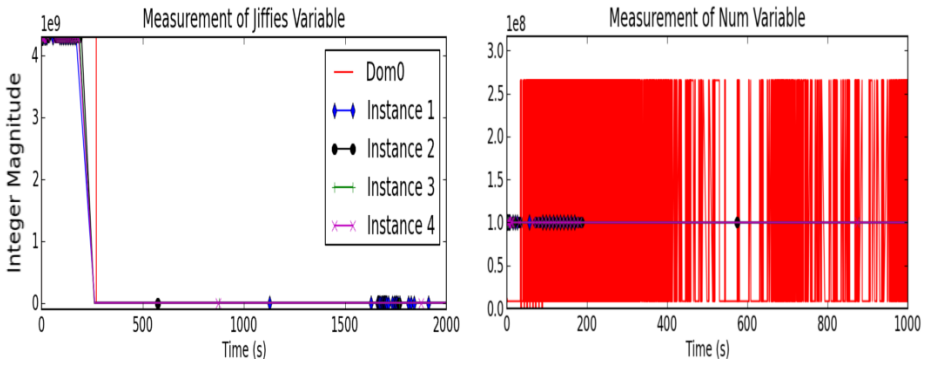

Fig. 4. Jiffie and num variables with 4 instances

The Cycle Variable. The cycle variable has a much more interesting, and entropy rich behavior across all instances, both guests and domain 0, and are shown in Figure 5. The guest instances show tight coupling between the cycle variable and each other, however the domain 0 instance is largely independent of any guest instance. In Table 1, the correlation and covariance of the cycle variable between the guest instance and the dom0 OS is rather low. The covariance for each pair is normalized to the first listed instance's covariance with itself. The correlation across guests is much higher than that of the cases with the domain 0 OS, and is shown in Table 2.

Table 1. Guest Instance Cycle contributions correlated with Dom 0 Cycle Contributions (4 guests)

Instance (with Dom 0)	Correlation	Covariance
DomU1	0.177657311	0.2019764114
DomU2	0.3175715104	0.2947513962
DomU3	0.4681160782	0.5268961915
DomU4	0.3753263986	0.3768116959

Table 2. Correlation and Covariance of Entropy Contributions Across Guest Instances

Instance Pair	Correlation	Covariance
2 Instances		
DomU1-2	0.7698154055	0.7775670479
4 Instances		
DomU1-2	0.6424939995	1.6548446143
DomU1-3	0.7283067151	0.7936474751
DomU1-4	0.893786794	2.2162510283
DomU2-3	0.8870338305	1.0030611841
DomU2-4	0.9862760451	0.9635573195
DomU3-4	0.8700133244	1.0424442764

6.2 Analysis of Results

The assumed ability to contribute to the dom0 entropy pool by generating interrupts on the guest instances turned out to be largely incorrect. While this makes the pool poisoning attack unlikely, the correlation between the guest instances was remarkably high over large runs of the samples. This violates the assumption made by the designers of the Linux RNG that the contributions are uniquely random; unknowable to anything but the RNG. The cycle variable is measured by the assembly instruction rdtsc. It is zeroed on a reset of the processor, and increases every clock cycle. The correlation is likely the result of the free running counter register being synchronized among cores, which is typical for multicore processors, though not guaranteed. So highly correlated guests were scheduled on the same CPU but a different core.

The general invariance of the rest of the two-thirds of each sample does negatively affect the entropy estimate of contributions to the pool. The num variable was invariant due to its source, the type of interrupt generating the event. For instance, a keyboard event would pass the keyboard scancode to the num variable. Being that guests receive all their I/O from the network, there is no variation. The invariance of the jiffie variable is most likely due from Xen trying to handle the jiffie clock for guests. Entropy generation rates are negatively affected by this. Therefore, the proposed Entropy Management System does serve a purpose to provide an emergency source of entropy.

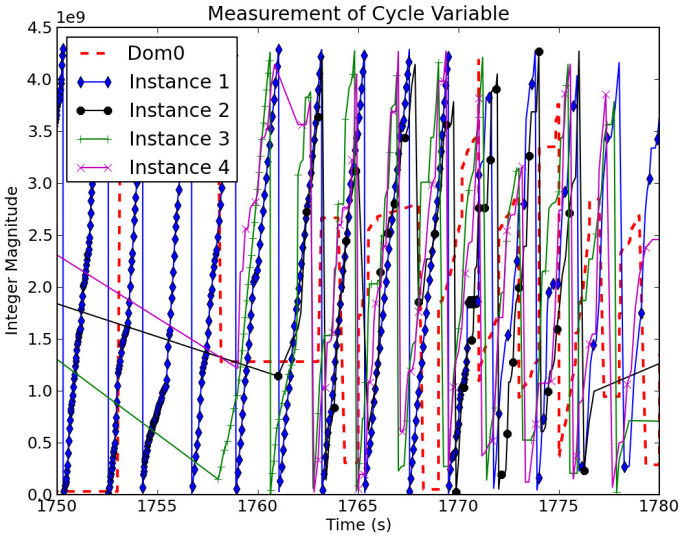

Fig. 5. Cycle variable with 4 instances

7 Discussions

Overall there aren't any glaring practical security vulnerabilities that are demonstrated in the Xen Cloud Platform, however the rate of recovery for guest instances is likely overestimated, which leads to the conclusion that the estimates were inflated throughout the tests. While this makes a theoretical cryptanalytic attack hypothetically possible, it is not much different from other hosts which suffer from overestimation. This is the reasoning behind the use of a dividing mechanism in the Entropy Management System. It's far better to underestimate the entropy than it is to overestimate it. If a guest instance is to be cloned, it ought to have its pool drained before cloning. Upon start up of the cloned instances, the pool should be again drained, and finally any keys should be regenerated. While the step of draining the pools after cloning was performed in tests, SSH keys were not regenerated, and as a result the RSA fingerprint for all instances was identical. Mechanisms for dealing with the shared state of cloned instances, namely breaking the common state for things that require uniqueness, would be a fruitful endeavor in securing cloud instances. Carelessness of users cannot be underestimated, and tools to automate these chores would be very useful. Another area which is hypothetically weak in respect to Cloud entropy is the loadbalancing mechanisms and cloning mechanisms. There are times when the instances may be loadbalanced over a public network. If an instance is cloned or loadbalanced across a public network, it would also be prudent to drain the pool, and regenerate keys.

8 Conclusions

In this work, we explored the potential weaknesses in the generation of random numbers in Cloud hosts, and provided tools to mitigate these weaknesses. First, the evidence of entropy coupling between domain U instances in Xen Cloud Platform hosts is revealed. There is a possibility of prediction of the variable given enough instances are under the control of an attacker.

Second, our experimental results show that virtualization affects entropy sample collection. The num variable in particular did not change once in any of the experiments, while the jiffie variable did exhibit one change throughout all the experiments, apparently most changes being filtered by the virtualization layer. Finally, the high correlation of the cycle variable is concerning, and makes a case for entropy gathering on Xen guests to be managed differently in the kernel.The use of the Cloud Entropy Management System sidesteps the problems with the correlation of samples by providing bailout entropy that is uncoupled. The implementation is low overhead, and consists of two daemons written in Python. It provides a reasonable way to ensure entropy estimates and entropy pools in Cloud guests aren't susceptible to exploitation.

References

1. Jun, B., Kocher, P.: The Intel Random Number Generator, Cryptography Research Inc., white paper prepared for Inter Corp., (April 1999), http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf
2. Taylor, G., Cox, G.: Digital randomness. IEEE Spectrum 48 (September 2011)
3. Lian, G.: Testing Primitive Polynomials for Generalized Feedback Shift Register Random Number Generators, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.318&rep=rep1&type=pdf
4. Kelsey, J., Schneier, B., Ferguson, N.: Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 13–33. Springer, Heidelberg (2000), http://www.schneier.com/paper-yarrow.ps.gz
5. Ferguson, N., Schneier, B.: Practical Cryptography, pp. 161–182. John Wiley & Sons (2003)
6. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Cryptanalytic Attacks on Pseudorandom Number Generators. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 168–188. Springer, Heidelberg (1998)
7. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the linux random number generator. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy. IEEE Computer Society (2006)
8. Mackall, M.: Linux Kernel Source 2.6.32.8 Random Character Driver, (/linux2.6.32.8/drivers/char/random.c in kernel source tree)
9. Beige, T.: Analysis of a strong Pseudo Random Number Generator by anatomizing Linux Random Number Device (November 2006), http://www.suse.de/~thomas/papers/random-analysis.pdf
10. Duda, K., Cheriton, D.: Borrowed-Virtual-Time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. In: Proceedings of the 17th ACM Symposium on Operating Systems Principles, SOSP 1999 (December 1999)

11. “Earliest deadline first scheduling” Internet: http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling (December 4, 2010) [April 26, 2011]
12. Mathai, J.: ”Scheduling - Xen Wiki” Internet: http://wiki.xensource.com/xenwiki/Scheduling (June 09, 2007) [May 7, 2011]
13. Park, S., Miller, K.: Random Number Generators: Good Ones Are Hard to Find. Communications of ACM 21(10) (October 1988)
14. LÈcuyer, P.: Efficient and Portable Combined Random Number Generators. Communications of the ACM 31(6), 742–774 (1988)
15. Carstensen, C., Fine, B., Rosenberger, G.: Abstract Algebra - Applications to Galois Theory, Algebraic Geometry and Cryptography. Heldermann Verlag (2011)

Security Modeling of Grid Systems Using Petri Nets

Peter D. Zegzhda, Dmitry P. Zegzhda, Maxim O. Kalinin,
and Artem S. Konoplev

Information Security Center, St. Petersburg Polytechnical University,
St. Petersburg, Russia
{zeg,dmitry,max,project}@ssl.stu.neva.ru

Abstract. The paper reviews the security problem with computing and information resources in Grid systems. It discusses security relative characteristics of Grid architecture and provides a common threat model of Grid. It summarizes methods being applied to improve security of Grid systems and discusses their disadvantages. There is proposed the Petri-net-based model of access control for Grid systems. That model enhances Grid security with trusted 'job' submission (in strict accordance with security policy constraints) and verification of the security implementation in Grid systems.

Keywords: Grid, information security, Petri net, security model, security policy, verification.

1 Introduction

Nowadays, Grid systems as kind of distributed computing systems have become a leading technology which is applied to solve work-intensive and resource-intensive tasks in scientific and commercial areas. Due to the higher value of information being processed by Grid hosts, Grid systems are focused on aspects of information security, specifically *computing and information resource protection*. This problem is caused by specific nature of distributed Grid systems, which are built basing on the principles of heterogeneity, common ownership of job processing infrastructure, high dynamics of states, and decentralization.

The processing of users' jobs is performed remotely on multiple host systems. But it takes effective methods to protect user data from host environment. Information security policies are to solve that problem using authorization regulations (usually, in form of constraints '*subject-object-rights*'). But due to specific nature of Grid systems there is no unified mathematical apparatus which would allow these requirements to be defined for each entity involved in job processing. Accounting of predefined relations is also required. This paper proposes a technology targeted to solve the problem of *computing and information resource protection*.

2 Background

Grid system resources are divided on several classes in accordance with user tasks:

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 299–308, 2012.

1. Computing resources. This Grid system is utilized by users to solve labor-intensive tasks which require allocation of significant amounts of CPU time or memory.
2. Storage resources. This Grid system is used as the user's information storage located on remote hosts.
3. Software resources. This Grid system is applied by users for data processing by applications which are unavailable in their own software environment.
4. Network resources. This kind of Grid is not directly available for users, but they are utilized for communications between other types of resources.

The underlying Grid system for the principle of common ownership of job processing infrastructure is implemented through the mechanism of virtual organization (VO), the dynamic community of users that share resources of the Grid system to solve a common tasks and in accordance with agreed rules [1]. It means that every user may consist in a few VO at once. Moreover, the VO's content changes extremely high.

Thus, the special nature of Grid systems, characterized by the properties of the decentralized, heterogeneous, and highly dynamic states, complicates the task of ensuring an adequate level of security protection of computing resources and user data. For example, for Grid systems it is difficult to use classical security models, and, consequently, to verify information security policies.

We distinguish the following classes of computer attacks which are typical for Grid systems (a common threat model of Grid systems is shown on Fig. 1):

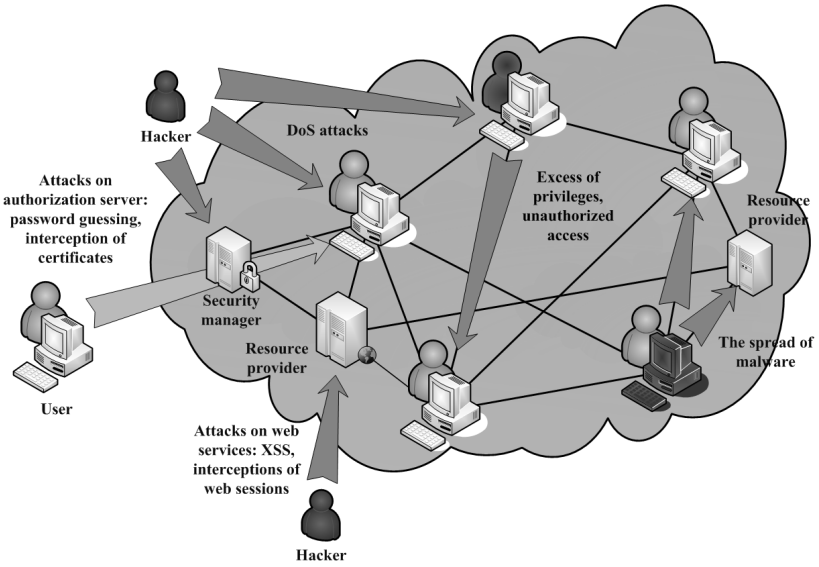

Fig. 1. A common threat model of Grid system

- **Denial of service** attacks carried out by:
  - network attacks targeted at basic services of Grid system (break functioning of the system as a whole);

  - enforcement of the authorized component to disconnect from the Grid system;
  - overload Grid system in which the work of users and services is difficult.
- **Spread of malicious software**. This type of attacks is especially important for Grid systems because of the ideal environment for its implementation (there is an open way for 'job' migration which can be used in insecure manner).
- **Unauthorized access to computing resources**. Attacks could come from both authorized users and components of Grid systems and an external intruder. We can distinguish the following subtypes of attacks of unauthorized access:
  - connection the unauthorized user or component to the system (component is supposed to be unauthorized if it has no certificate issued by the trusted certification center);
  - attempt to access user's data by the user's processes of the host environment;
  - attempt to exceed the privileges of user, application, or service in Grid system.

In the next section we consider the related approaches applied to protect Grid system. Section 4 presents a mathematical apparatus of Petri nets for modeling of Grid system security. The specified technique allows trusted 'job' submission (in accordance with security policy constraints) and verification of the security implementation in Grid systems. The conclusion summarizes the main results of our work. The paper thus provides a solution of computing and information resource protection problem.

3 Related Works

To provide protection against denial of service attacks and the spread of malicious software, Grid systems implement special hardware and software components (security managers) with intrusion detection systems (IDS), firewalls, and antivirus agents installed on them [2]. There are also several research aimed at solving the problem of anomaly detection in distributed computing systems [3].

Security managers are integrated with dedicated communication channels, which in the case of intrusion detection alerts are broadcast. Receiving such notification, each host duplicates it to all resource providers being connected to it. As a result, all hosts isolate the problematic host. It thus prevents the possibility of attacks spreading in Grid systems.

In addition, in some Grid systems the fuzzy logic of trust is used [4]. Each host is initially labeled. This label shows the trust level assigned to it by other components of Grid system. If attack from that host is fixed, the trust level is decreased. While search for a suitable host for a user-defined jobs, the hosts with the highest trust level are chosen for running these jobs.

In contemporary Grid systems (e.g., Globus Toolkit [5], UNICORE [6], gLite [7], Gridbus [8], and BOINC [9]), it is distinguished two major security mechanisms: authorization and authentication [10]. There are two approaches of user authorization. The first one is based on a special service – GRAM (Grid Resource Allocation and Management). Using GRAM, the user can delegate their applications to the resource provider, and then get the results from it [11]. The second one is taken from cloud computing systems and is based on the web services. There are some services running on the resource provider. Each service is "attached" with operations. User can get access to the operations attached to the required services.

Since platform for Grid systems is a set of personal computers, an important task for Grid is to ensure confidentiality and integrity of user data. There is required protection of user-related data from the local host users including 'root' user. For this purpose, the trusted software and hardware platform is used on the hosts. User data are protected in a special encrypted repository at host environment.

4 Security Modeling of Grid Systems Using Petri Nets

Compliance with the rules of information security policies are guaranteed by the access control. The high heterogeneity of Grid systems, multiple user authentication mechanisms and lack of a centralized security server make it difficult to use classical security models. In [12] there is presented a unified model which allows logical specification of security policies in Grid systems taking into account the different mechanisms of user authentication. But there Grid system is considered as a static set of the system states, which either meet the requirements of information security policies or not. Since each user of Grid system may at any time belong to several VOs, to close threats of unauthorized access it should be able to establish the relations of division between all potential members of the computational process. Therefore, according to the presented threat model problem of unauthorized access is still open for Grid systems.

We propose an approach to protect Grid system resources against unauthorized access based on secure job submission in accordance with the requirements of security policies. Implementation of this approach involves the mathematical apparatus of functional colored Petri nets. In contrast with access control models describing the time and change of state in considerable system as two continuous variables, Petri net is one of several mathematical representations of dynamic discrete systems. Another great advantage of chosen mathematical apparatus is possibility of describing the predefined access relations in Grid system states.

The finite set of vertices $M = \{m_i\}$ of the graph $c = (M, T, *)$ are the nodes of the Grid system (hosts, resource managers, etc.), where $T = \{t_i\}$ is a set of transitions between the vertices of *c*. Markers denote the requests for a particular type of Grid system resource. The transitions are implemented by the function that takes into account time spent in the queue tags (i.e., the job processing waiting time).

Consider the basic representation of Grid system in the form of Petri net (Fig. 2). There are four resource providers (M_1, M_2, M_3 and M_4). Generally, a set of users *U* and a set of vertices of *M* cannot coincide. This is explained by the fact that the same node of Grid system can operate with multiple users. On the other hand, some nodes may not be authorized by any user, even though the computing power of this unit is part of the resources of the Grid system. However, for simplicity here and further it is assumed that each resource provider meets single Grid system user. Let's assume that all user requests go through the resource provider M_2, and any job can be initiated only by user U_1 (on M_1 resource provider). Moreover, the specified job can always be performed by the other two host systems M_3 and M_4. Consider job submission in the specified example.

Suppose that t_i — transition probability of the marker from one node of Grid system to another. In our example $t_0 = 1$, since there is only one connection to resource provider M_2 in the system and all job requests pass through it. t_1, t_2 — the probabilities that computing resources of M_3 and M_4 nodes are required to perform the job. t_3 — the probability that computing resources of both M_3 and M_4 are required to perform the specified job. Such situation is possible, for example, when two programs should be executed to get job solved, and each of them is installed on only one host. t_4 and t_5 — the probabilities that the task has been successfully solved on the appropriate resource provider, and its result is returned to the user U_1. The initial marking of the net is presented with the vector $\mu = (\mu(M_1),...,\mu(M_n))$, where n — a number of Petri net nodes. For the net shown in Fig. 2, μ = (1, 0, 0, 0).

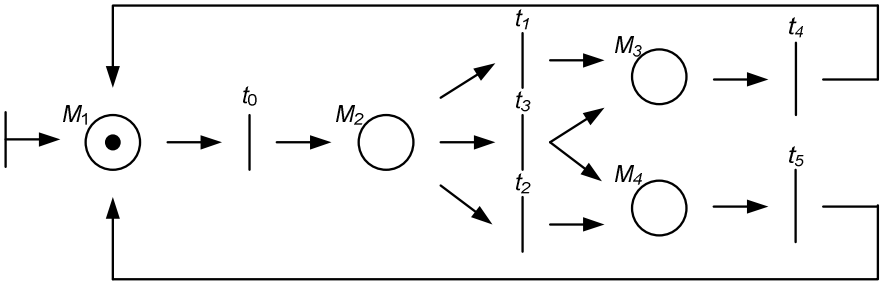

Fig. 2. An example of representation of Grid system using Petri net

Real Grid system has a more complicated architecture with several resource providers each of which has a particular set of connected hosts (Fig. 3). In addition resource providers can move job requests between themselves (transitions t_{31} and t_{32}). It happens in two cases:

1. The limit of active job requests is reached for that resource provider. In that case the load balancing between resource providers is performed, and a part of requests goes to other resource providers.
2. There are no available hosts connected to the specified resource providers which have an appropriate type or count of computing resources. Therefore, a part of job requests go to another more appropriate resource provider.

In considered Grid system implementation, there are no limits associated with security policy requirements. To take into account that limits, let's define the set of limits (the access rights in terms of information security) *Rights,* defined as constraints in security policy, and the set of predefined relations, i.e. the set of marks (the job requests of users) which are presented on the specified node at the moment. We define the type of mark as a cortege $T =< F, A >$, where F — a class of resources requested by user for his job (computing resources, storage resources, etc.), A — the security attributes using to verify compliance with the regulations of security policy.

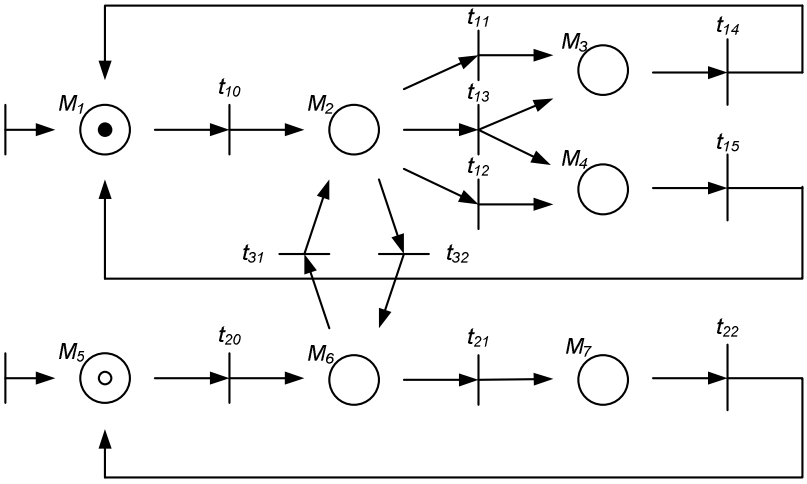

Fig. 3. Classification of resources and nodes in Grid system

For example, on Fig. 3 there are presented marks corresponding to the user's requests of two classes on vertices M_1 and M_5 correspondently. As a security attributes appear subjects (the Grid system users) access IDs, who had initiated specified request, $U \subseteq A$. In relation to the graph vertices cortege T defines the type of Grid system node.

Movement of user's job request from the node m_i to the node m_j means that the conditions for transition t_{ij} triggering are occurred. Finally it is possible to declare conditions of trusted job submission:

- User and resource provider trust each other, i.e. have successfully passed authentication procedure.
- There is a predefined user account on the specified host which corresponds to the user initiated the job. All calculations will be performed with privileges of that user account.
- Requested resource provider is available and has an appropriate type.
- Takes into account predefined access relations on the specified resource provider:
  - the requested access type is equal to all existing access types on the specified resource provider or
  - there is at least one existing access type on the specified resource provider that is not equal to the requested access type but security policy won't be violated if the requested access type grants.

Now we can define an operator which acts on the set of transitions $\{t_{ij}\}$:

$$\Psi : \langle J, M, Rights, Relations \rangle \rightarrow \{True, False\}$$

where:

- J is a set of all jobs in the Grid (note that each job is uniquely corresponds to the user that initiated the job);

- *M* is a set of resource providers presented in Grid system;
- *Rights* is the access rights in terms of information security policy;
- *Relations* is set of predefined access relations.

Thus, while choosing a suitable resource provider for the specified job there are considering not only availability and type of resource providers but also the requirements of information security policies that allow or prohibit the use of resources on the specified node by the user that initiated the request.

Applying predicate logic allows considering parameters that define the domain of the operator as variational. The formalization of operator Ψ, as well as the presence or absence of its parameters and the range of values of the operator generate a set of practical tasks aimed at enhancing security for Grid systems:

1. Trusted distribution of job requests. Let the sets *U*, *M*, *Rights*, *Relations* be defined. The presence of these parameters allows building the algorithm of host systems search to run the query specified by the user, taking into account the requirements of security policies. Thus, the necessary condition holds the secure distribution of user job requests for the provision of Grid system resources — every state of the Grid system meets the constraints of security policy.
2. Security policy verification. Let the sets *U*, *M*, *Rights* and range of values of operator Ψ be defined. The presence of these parameters allows solving the problem of identifying predefined relations leading to a violation of the security requirements.
3. Creating of security template settings. Let the sets *U*, *M*, *Relations* and range of values of operator Ψ be defined. The presence of these parameters allows creating security settings, which can then be used to automate the process of setting up of Grid system security configuration in accordance with the requirements of security policies.
4. Finding intruders (users who violate the security constraints). Let the sets *M*, *Rights* and *Relations* be defined. The presence of these parameters allows identifying of the users that perform actions which lead to violation of the security rules.

Consider job submission procedure using following example based on Globus Toolkit implementation of Grid. Suppose we have seven resource providers connected as shown on Fig. 3. For each resource provider, there are defined the following types: T_1 = {'*storage resources*'}, T_3 = {'*computing resources*', '*storage resources*'}, T_4 = {'*computing resources*}, T_5 = {'*storage resources*'}, T_7 = {'*storage resources*'}.
Also suppose the security policy defined using two types of access (storage access and computing access). It includes the following access constraints:

1. User U_1 *can store* data *on* host M_3.
2. User U_3 *can execute* applications *on* host M_7.
3. User U_2 *can execute* applications *on* host M_3.
4. User U_4 *can store* data *on* host M_1.
5. User U_2 *cannot do anything on* host M_3.

Initially there are no any jobs processing on hosts of Grid system. Suppose that user U_1 on host M_1 create job J_1 with the following attribute (in terms of Petri net): type of mark T = <'*Storage resources*', U_1>. Then J_1 goes to resource provider M_2 using t_{10}

transition arc. Transition t_{10} is triggered unconditionally because J_1 came from the host that had created the given job, and obviously the host U_1 could not process that job. As soon as J_1 appears on M_2, the resource provider starts a procedure of searching for the mostly appropriate host for job processing. It knows a list of hosts linked with it, their state, and a type of the resources they can provide. The host M_3 has '*ready*' state and the required type for J_1 processing ('*storage resources'*). On the next step, M_2 checks that the job submission meets the requirements of current security policy. It finds the rule 1 (from the security constraints listed above) that allows U_1 to store data on the host M_3. Therefore, J_1 appears on M_3 (Fig. 4). Then, let's suppose that J_2 with the type $T = <$'*computing resources*', $U_2>$ appears in Grid system. It means that J_2 came from the host M_5 and moved to the resource provider M_6. As there is no host connected to M_6 which has the appropriate type, M_6 moves J_2 to M_2. Finally, J_2 appears on M_4 because it also supports '*computing resources*' type and there is no security constraint that prohibits '*execute applications*' access on M_4 for the user U_2.

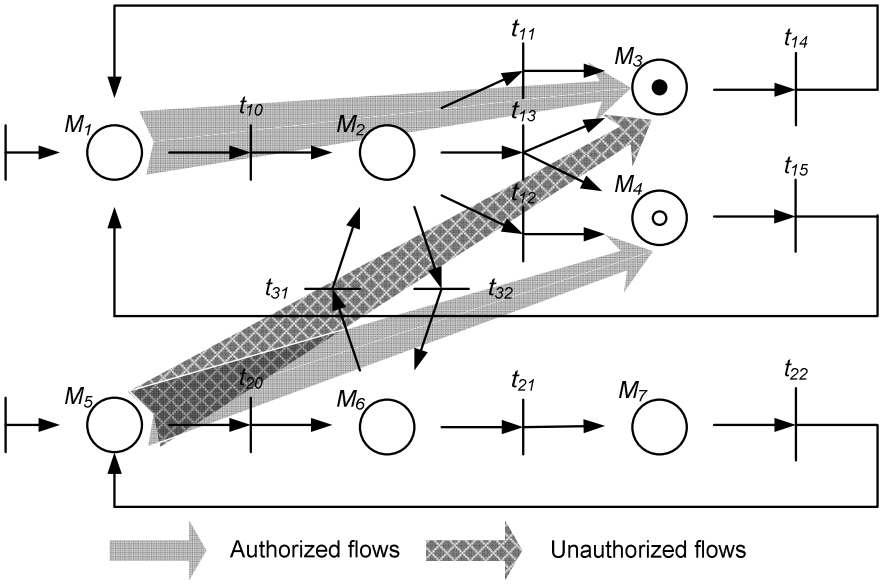

Fig. 4. Job submission procedure

Seems that job submission procedure was performed in agreement with requirements of the specified security policy but more detailed examination shows that it's not true. At the end of job submission procedure, we can see that both users U_1 and U_2 have access to host M_3. In addition, some application of the user U_2 is running on the host M_3 while the user U_1 stores his data on the host M_3. In other words, the user U_2 has access to the user's U_1 data that is not allowed by security policy. The unauthorized flow of job submission is shown on Fig. 4 with dashed arrows.

Now consider a specified procedure but using our approach of trusted job submission. First part of procedure will be the same till the stage when J_1 appears on the host

M_3. When J_2 comes to M_2, the resource provider in spite of M_3 is most appropriate host for J_2 processing t_{11} transition wont trigger because of predefined relations on the host M_3 (the user U_1 has already had access to M_3). That is why J_2 at the end comes to the host M_4 which is also available, has appropriate type and no predefined relations at all. The trusted flows of job submission are shown on Fig. 4 with solid arrows.

The task of finding hosts that are suitable for job submission procedure is performed by special service of Grid system, which is running on the resource providers. Therefore, to ensure the trusted job submission, the operator Ψ is to be integrated into the specified service. Thus, when choosing a suitable host will be considered not only their availability and the type of Grid system resources, but also the requirements of security policies with accounting of predefined relations in host environment.

5 Conclusion

The paper has addressed to the problem of computing and information resource protection in Grid systems. It provided threat model of Grid systems and disadvantages of existing methods to improve security of Grid systems. There is presented a theoretical model of access control for Grid systems based on Petri nets. It allows the trusted job submission procedure in accordance with the requirements of information security policies. Novelty of the proposed model is describing the time and change of state in Grid systems as two discrete variables and taking into account predefined access relations. Solving the specified problems as well as implementation of the suggested approach will allow to automate the process of safety analysis, and thus provide a high level of security in Grid systems.

References

1. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, 2nd edn (2004)
2. Lohr, H., Ramasamy, H.V., Sadeghi, A., Schulz, S., Schunter, M., Stuble, C.: Enhancing Grid Security Using Trusted Virtualization. Springer (2007)
3. Stepanova, T., Zegzhda, D., Kalinin, M., Baranov, P.: Mobile Anomaly Detector Module Based on Power Consumption Analysis. In: The 2010 International Conference on Information Security and Privacy (ISP 2010), Orlando, FL, USA, July 12-14 (2010)
4. Song, S., Hwang, K., Macwan, M.: Fuzzy Trust Integration for Security Enforcement in Grid Computing. Springer (2004)
5. The Globus Security Team. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective / The Globus Security Team (2005), `http://globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf`
6. Internet resource: `http://www.unicore.eu`
7. Sciaba, A., Burke, S., Campana, S., Lanciotti, E., Litmaath, M., Lorenzo, P.M., Miccio, V., Nater, C., Santinelli, R.: GLite 3.2 User Guide. – CERN (2011)
8. Buyya, R., Venugopal, S.: The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. In: 1st IEEE International Workshop on Grid Economics and Business Models, Seoul, Korea, April 23 (2004)

9. Internet resource: http://boinc.berkeley.edu
10. Kalinin, M., Konoplev, A., Markov, Y.: Control of the security policies requirements in grid-systems. In: Proc. of the Conference Information Security of Russian Regions (ISRR 2011). St. Petersburg, Russia (2011)
11. Alfieri, R., Cecchini, R., Ciaschini, V., Dell'Agnello, L., Frohner, A., Gianoli, A., Lorentey, K., Spataro, F.: VOMS, an Authorization System for Virtual Organizations. LNCS (2003)
12. Kalinin, M., Markov, Y.: Verification of security policies requirements in grid-systems. Information Security Problems. Computing systems 2 (2011)

Using Graph Theory for Cloud System Security Modeling

Peter D. Zegzhda, Dmitry P. Zegzhda,
and Alexey V. Nikolskiy

Information Security Center, St. Petersburg Polytechnical University, St. Petersburg, Russia
zeg@ssl.stu.neva.ru

Abstract. The paper discusses the security problems of cloud systems. It also contains a model of cloud systems that allows formally describe different security problems. The proposed model is based on graph theory and it describes main features of virtual machines in cloud systems. The paper formally presents a transformation of data operations that happens in hypervisor software due to virtualization technology. It allows formally define several cloud system security problems of hypervisor software. The paper also contains a discussion about other security problems with shared virtual machines in the cloud.

Keywords: cloud, information security, virtualization, model, graph theory.

1 Introduction

Cloud computing is an intensively developed modern business model that is embedded worldwide. Cloud computing service user performs required computations using temporary allocated resources in cloud but user can't control internal cloud architecture where actual resources are located. This idea produces main benefits and hazard of cloud computing technology.

Using cloud computing user data is moved to cloud internals with loss of user control over this data. In addition cloud service providers practically have no liability for user data security [13]. It also considerably increases the complexity of security systems development and deployment in cloud and requires revision and adapts of existing security models to new cloud reality.

To be able to create a secure cloud system it is required to create a security model, allows describing main features of virtual machines in cloud environment. The main features of virtual machines in cloud are:

- virtual machine application as a network node in the cloud;
- virtual machine migration;
- virtual machine replication;
- virtual machine image holding in shared for all network nodes storage server;
- consolidation of virtual machines in isolated virtual networks.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2012, LNCS 7531, pp. 309–318, 2012.

2 Related Works

Continuous development of cloud systems produces a lot of different mathematical models for many features of cloud. Joe Weinman in his paper [8] has created an axiomatic for the cloud theory and a mathematical model based on this axiomatic. His work describes one of the most generic and accurate model for cloud computing. The cloud in his work is defined as a structure that must satisfies five formal axioms: it must be 1) Common, 2) Location-independent, 3) Online, 4) Utility, and 5) on-Demand. One of the main parts of this cloud structure is a graph that changes in time. This model is not appropriate for the virtual machine feature modeling and security cloud modeling because it is very generic and doesn't define explicitly security subjects and objects for access operations.

In our previous paper [14] we have provided a review of main security problems and threats for any cloud system. The paper [13] trends to focus on the discussion about security problems of virtualization software and its application in cloud based information systems. But in this paper authors presents a formal model for the cloud and a formal definition for virtualization software security problems.

W.K. Chan, Lijun Mei and Zhenyu Zhang in their work [9] have presented a model based on graph theory suitable to describe application behavior in the cloud systems. The article is focused on challenges with testing of application in the cloud environment. Purposed model represents computing resources on nodes with different attributes. The most interesting part of this model's ability is representing a usage of any resource as a predicate on an edge of the graph. There is no virtual machine entity explicitly presented in this model, so it is not applicable for security model that is required for virtual machine features description.

Yingmin Li and Omar Boucelma in their paper [10] have presented a model based diagnosis approach to monitor workflow provenance in the cloud. The article is focused on diagnosis of security and verification approach based on modeling of cloud systems. The model in Yingmin and Omar work is based on Petri Nets. This approach and model is not enough to model cloud system security and virtual machine features.

Hui et al. in their article [11] extends the formal model of Abstract State Services (AS2s) by formalizing the notion of plot of a service. This approach is very interesting, because it allows to actually capture the possible sequencing of service operations, which is only implicitly present in the AS2 model. This work is focuses only on service side of cloud systems and doesn't allow to model network communications within cloud structure.

Thomas et al. [12] has created not only a formal model of cloud systems but also a simulation framework and a simulator. The cloud model in their work is also based on the graph where each node corresponds to a computing entity like a physical or a virtual machine and each edge is a communication link between two nodes. This approach allows to model network communications in cloud systems but it is not applicable for security modeling.

It is also important to mention that it is possible to describe cloud internal network nodes with RBAC model [15], but cloud system security problems has more specifics then just roles of software agents within network system. For example, virtualization

software adds more challenges to existing security systems, network topology has not only real wired connections, but also virtual isolated and shared networks. All these features of cloud systems are complicated to be represented in RBAC model.

It is useful to exercise graph theory for cloud system modeling that consists of many network nodes and computational resources. Graph models can be used to formally describe security requirements for complex network systems like grids [16]. But to create a security model for cloud systems and virtual machine features it is required to include security entities and virtual machine features in this model explicitly.

3 Virtual Machine Security Model for Cloud System

Basically any cloud system is a complicated computer network that can be named an internal cloud network [1]. Any cloud is based on cloud platform software (like VMware vSphere or Xen Cloud platform) that defines all rules and mechanics of internal cloud network nodes cooperation and data exchange. Modern data processing centers running cloud platform software can consist of thousands of network nodes. Any node execute specified role in internal cloud network and it is possible to mark some major set of roles $\{\mathrm{VmHost,Storage,Controller}\} \subseteq \mathrm{R}$ that exists in any cloud system, where: VmHost – This role is assigned to Working Nodes; $\mathrm{Storage}$ – This role is assigned to cloud Storage nodes; $\mathrm{Controller}$ – This role is assigned to all Controller nodes in the cloud.

Due to the fact that any internal cloud infrastructure is a kind of local area network, it is possible to represent cloud infrastructure as an undirected graph $C_L = (N, \eta)$, where $N = \{n_i\}$ – a set of network nodes from internal cloud network and $\eta \subseteq \mathrm{N} \times \mathrm{N}$ - a set of edges between network nodes that represents an allowable network links between them. η is defined by physical connection, individual firewall and router settings. This graph is undirected because data exchange in the system network.

It is possible to define a function $Role$, that associate any node from the graph with a set of roles that are assigned to this node: $Role : N \to P(R)$. In huge data processing centers any node has only one role ($\forall n \in N, \left|Role(n)\right| = 1$) [2], but cloud platform software allows to host multiple roles on single node.

Let $V = \{v_i\}$ defines a set of all virtual machines in the cloud. It is also possible to define another undirected graph $C_V = (V, \upsilon)$, where $\upsilon \subseteq \mathrm{V} \times \mathrm{V}$ is a set of edges between virtual machines that represents an allowable virtual network links.

Let $M = \mathrm{V} \cup N = \{m_i\}$ define a set of all machines in the cloud (joining all virtual machines and all network nodes from internal cloud infrastructure).

Finally it is possible to define a cloud graph $C = (M, c)$, where $c \subseteq \mathrm{M} \times \mathrm{M}$ is a set of edges between machines that represents an allowable network links. For c it is also true: $c \subseteq \eta \cup \upsilon$.

Beyond network relations represented by graph C, set of virtual machines V and set of cloud internal network nodes N has some relations that produces by virtualization technology features that makes possible to define extra relations between V and N:

- $H \subseteq \mathrm{V} \times N$ – a set of tuples (v_i, h_i) where v_i is a virtual machine instance running on node h_i;
- $I \subseteq \mathrm{V} \times P(N)$ – this is a set of tuples (v_i, S_i) where v_i is a virtual machine instance and S_i is a set of all nodes of internal cloud network that has image of this virtual machine instance;
- $F \subseteq \mathrm{V} \times P(O)$ – this is a set of tuples (v_i, O_{v_i}) where v_i is a virtual machine instance and O_{v_i} is a set of all files (as a subset of all objects in the system O) that represents virtual machine image in the cloud;
- $R \subseteq \mathrm{V} \times P(N)$ – this is a set of tuples (v_i, T_i) where v_i is a virtual machine instance and T_i is a set of all nodes of internal cloud network that replicates running state of this virtual machine instance.

The software that is running on the machine m_i may be represented as a set of programs $P_{m_i} = \{p_j^{m_i}\}$. Let $P = \{p_i\}$ define a set of all programs in the cloud $P \subseteq \bigcup P_{m_i}$. Let $Soft \subseteq M \times P(P)$ defines a set of tuples that links together all machines in the cloud with programs set installed on them. It is also possible to define: $Soft^N \subseteq N \times P(P) \subset Soft$ for network nodes; $Soft^V \subseteq V \times P(P) \subset Soft$ for virtual machines.

Each virtual machine is a software abstraction that can be implemented by using hypervisor software. This means that each running virtual machine is a program within software of some network node in cloud:

$$\forall (v_i, h_j) \in H, v_i \in V, h_j \in N : \exists P_{n_j} \subset P, (n_j, P_{n_j}) \in Soft^N \wedge v_i \in P_{n_j}$$

Finally as for any information system for security purposes it is possible to define more statements [3]: $U = \{u_i\}$ and $S = \{s_i\}$ be a set of users and subjects; $Id \subseteq U \times P(S)$ - the identification set; $\mathrm{Im} \subseteq P \times S$ - impersonation set.

Now we can define a cloud system state as a tuple: $\psi = (C, P, U, S, H, I, R, Soft, Id, \mathrm{Im})$.

where: C - is a network graph of all machines in the cloud including virtual machines and cloud internal network nodes; P - is a set of all programs in the cloud that is running on machines or stored in some images; U - is a set of all cloud users; S - is a set of all subjects in the cloud that are defined by the software running in the cloud; $H, I, R, Soft, Id, \mathrm{Im}$ - are set of tuples that defines current state of different relations between cloud users, subjects, machines and programs.

In this case cloud system can be represented as a finite automation $\Omega = (\Psi, \psi_0, \tau)$, where Ψ defines a set of all states of cloud, $\psi_0 \in \Psi$ is an initial state of the cloud and $\tau : \Psi \to \Psi$ is a state-transition function for the cloud system.

Live migration [4] can be represented in purposed model like an operation that changes cloud state in the parameter H. It means that this operation moves system from state ψ to state $\tilde{\psi}$ and:

$$H^{\tilde{\psi}} = H^{\psi} \setminus (v_i, n_j) \cup (v_i, n_k), v_i \in V^{\psi}, n_j \in N^{\psi}, n_k \in N^{\psi}$$

$$\psi = (C^{\psi}, P^{\psi}, U^{\psi}, S^{\psi}, H^{\psi}, I^{\psi}, R^{\psi}, Soft^{\psi}, Id^{\psi}, \mathrm{Im}^{\psi})$$

$$\tilde{\psi} = (C^{\tilde{\psi}}, P^{\tilde{\psi}}, U^{\tilde{\psi}}, S^{\tilde{\psi}}, H^{\tilde{\psi}}, I^{\tilde{\psi}}, R^{\tilde{\psi}}, Soft^{\tilde{\psi}}, Id^{\tilde{\psi}}, \mathrm{Im}^{\tilde{\psi}})$$

Finally a proposed model allows describing all basic features of virtual machines in cloud systems, presented in the beginning of this article:

- virtual machine application as a network node in the cloud can be described this way: $V \subseteq M$;
- virtual machine migration can be described this way:
- $H^{\tilde{\psi}} = H^{\psi} \setminus (v_i, n_j) \cup (v_i, n_k), v_i \in V^{\psi}, n_j \in N^{\psi}, n_k \in N^{\psi}$;
- virtual machine replication in purposed model presented this way: $R \subseteq \mathrm{V} \times P(N)$;
- separation of virtual machine image holding and virtual machine execution can be described this way: $\forall (v_i, S_i) \in I, \exists n_j \in S_i : VmHost \in Role(n_j)$;
- consolidation of virtual machines in isolated virtual networks presented in purposed model as a separated graph C_v.

Putting virtual machines and network nodes in single graph allows graph theory using for deep analysis of whole network system of the cloud. It is also possible to formally define predicates (for example: $\forall v_j \in V : \exists n_i \subset N : (v_j, n_i) \in c$) that should be enforced by security systems in the cloud.

4 Model Application Experiment

Authors has created the graph for internal network of the cloud system in St. Petersburg Polytechnical University. This cloud system is shared between different

departments of the university, including Technical Cybernetics Department. The formal graph is based on explained model and includes 112 machines (24 hosts and 88 virtual machines). The graph was built with special automation software we developed based on network scanning utility. The sub graph (Fig. 1) of this graph was analysed and finally several issues was found. For example, the node N_1 has a combination of two roles (a Storage and Controller) and single authentication mechanism and administrative users for both services. Also a virtual machine V_{43} was used as a virtual gateway and DHCP server for isolated network of two other virtual machines (V_{41}, V_{42}) that was owned by one of the lab of Technical Cybernetics Department. Experiments shows that machine V_{43} has not only a direct link to the main router of the cloud (N_4), but also links to other working nodes in the cloud (bold lines on the Fig. 1), which is a security issue, that should be solved. As a result of these experiments authors of this paper has created a report with 11 security issues found in the cloud system. The security issues was discussed and solved with head engineers of this cloud system.

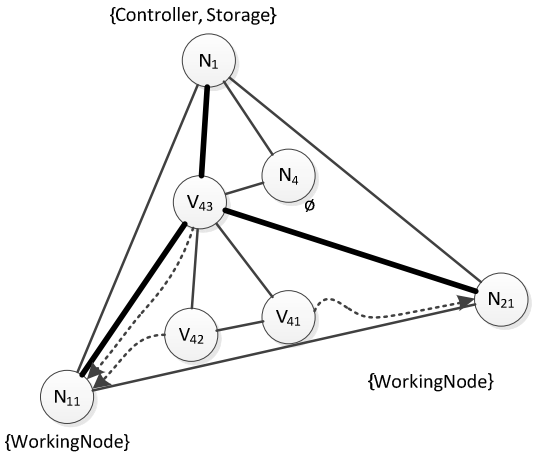

Fig. 1. University cloud graph segment

5 Hypervisor Security Problems in Cloud

As described above each virtual machine instance in cloud is a program and in addition each virtual machine instance also has some programs running on it:

$$\begin{cases} \mathrm{V} \subseteq \mathrm{P} \\ \forall v_j \in V : \exists P_{v_j} \subset P, (v_j, P_{v_j}) \in Soft \end{cases}$$

These two statements produce next one:

$$\forall (v_i, n_j) \in H, v_i \in V, n_j \in N, (v_i, P_{v_i}) \in Soft, p_{v_i} \in P_{v_i} :$$
$$\exists (u, S_u) \in \mathrm{Im}, u \in U, S_u \subset S, \{s_v, s_n\} \subset S_u, (v_i, s_n) \in Id \wedge (p_{v_i}, s_v) \in Id$$

which means that for some cloud user u it is possible to define at least two potentially different subjects: s_v - a subject that impersonates program p_{v_i}, running in virtual machine; s_n - a subject that impersonates virtual machine v_i, running on host n_j.

When the program p_{v_i} running inside virtual machine v_i performs write operation op^v with file f located on the hard drive of this virtual machine program p_{v_i} have to use an operating system software running in the same virtual machine v_i. If operation op^v is allowed then operating system has to perform a set of low-level operations $\{op_i^v\}$ to write some data on the hard drive device (Fig. 2). Basically each operation op_i^v can be either IO-ports manipulations or DMA transfers handling. An object parameter of these operations can be IO ports or memory regions. All of these operations are performed by operating system software at the highest privilege level inside virtual machine (not a s_v subject). Next each of op_i^v operation should be transformed to some other operations op_i^h by virtualization software [5]. Type of each operation op_i^h and a resource is depended on virtualization software implementation and virtual machine configuration.

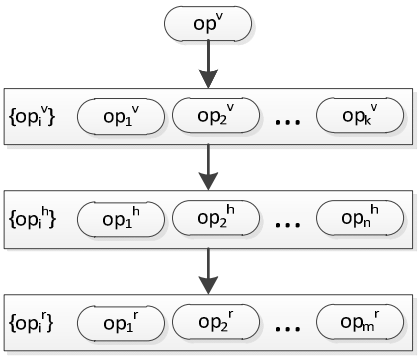

Fig. 2. Data access operation transformation

In case when virtual hard drive for virtual machine is located in the remote Storage n_r - is a host, where hard drive image files are stored (presented with set of files $\{f_i^v\} \subseteq O$). It means that set of operations $\{op_i^h\}$ actually is a set of network

communication data exchange operations using edge $(n_i, n_r) \in \eta$ in the graph C_L. It also means that on the host n_r there is another set of operations $\{op_i^{\ r}\}$ performs actual access to virtual machine image data (Fig. 2).

The example above produces the situation where cloud system user u associates with at least three different subjects that uses in three different software programs, all of these programs are running in three different execution environments (Table. 1).

Table 1. File access operation hierarchy

Execution environment	Software	Operations	Object	Object identifier	Subject
v_i	p_v	op^v	f	The name of the file	s_v
v_i	OS kernel inside virtual machine	$\{op_i^{\ v}\}$	hard drive sectors	sector numbers	OS kernel
n_i	v_i	$\{op_i^{\ h}\}$	$\tilde{f}$ - socket	a socket number associated with storage server	s_n
n_i	hypervisor	IO operations with real network device	low-level IO areas	network address	hypervisor
n_r	p_r	$\{op_i^{\ r}\}$	$\{f_i^{\ v}\}$	image file names	s_r
n_r	OS kernel on the storage server	IO operations with real hard drive device	hard drive sectors	sector numbers	OS kernel

This may produce several security problems that can be exercised by attackers [14].

In the first case if cloud system is used like a SaaS service by the user then it is possible that two users perform operations within single virtual machine but this virtual machine performs actual operations using single subject rights in virtualization software. It means that multiple users inside virtual machine may be associated with one subject on the host system, that is not really belongs to any of that users:

$$\exists(u1, S_{u1}) \in \mathrm{Im}, (u2, S_{u2}) \in \mathrm{Im}, u1, u2 \in U, S_{u1}, S_{u2} \subset S, s_{v1} \in S_{u1}, s_{v2} \in S_{u2},$$
$$\{(p_{v1}, s_{v1}), (p_{v2}, s_{v2})\} \subset Id, (v_i, P_{v_i}) \in Soft, \{p_{v1}, p_{v2}\} \in P_{v_i}$$
$$(v_i, s_n) \in Id, s_n \notin S_{u1} \wedge s_n \notin S_{u2}$$

Second situation may happen because emost of virtualization software implementations like Xen or KVM implements virtual machine using a set of components that runs using different subjects. For example, in Xen any virtual machine runs with common unprivileged mode, but to perform some operations virtual machine use a special qemu process that runs with administrator privileges at the host system [6]. It means that some operations of users in virtual machine finally performed by administrator user on the host system. This problem allows attackers to perform very dangerous actions with administrator privileges on the host [7]. This real issue in terms of purposed model can be presented this way:

$$\exists(u1, S_{u1}) \in \mathrm{Im}, u1 \in U, S_{u1} \subset S, s_{v1} \in S_{u1}$$
$$(p_{v1}, s_{v1}) \in Id, (v_i, P_{v_i}) \in Soft,$$
$$(v_i, s_n) \in Id, s_n \in S_{u1}$$

Both cases means that if some subject inside virtual machine has rights to virtual resources from virtual machine image then access control for these resources exists only in operating system running inside virtual machine but actual data access operations are performed in some other execution environment with no security control.

Looking at hypervisor software with operation set translation formalism can also be used for user behavior anomaly detection [17] in future works.

It is possible to declare that to make secure use of virtualization in the cloud any user of cloud system should be represented inside virtual machines only by subjects that has the same rights and privileges as this user in whole cloud system.

Purposed model allows to formally describing conditions of attacks in the cloud system, like described above.

6 Conclusions

Virtualization software implementation makes huge impact on overall cloud system security. This software makes transformation of one data access operation to the set of other data access operations with changing objects, subjects and even types of the original operation. Proposed mathematical model for cloud architecture allows describing basic entities and operations in cloud internals and it shows if some subject inside virtual machine has rights to some virtual resources from virtual machine image then attacker may get access to whole virtual machine image data.

This paper contributes a formal model for cloud systems that allows to formally describing of conditions for attacks performing in the cloud system. It also contributes a condition for secure cloud systems. Conditions may include users and subject relations, real and virtual networking topology, also based on the node role, conditions on software and virtual machine hosting.

References

1. Catteddu, D., Hogben, G.: Cloud Computing. In: Benefits, Risks and Recommendations for Information Security / European Network and Information Security Agency, ENISA (November 2009)
2. Michael, H.R.: VMware vSphere in the Enterprise (July 28, 2009), http://www.hypervisor.com
3. Zegzhda, P.D., Zegzhda, D.P.: Dynamic security methodology / MaBIT conference materials
4. Clark, C.: Live Migration of Virtual Machines. University of Cambridge Computer Laboratory Cambridge, UK, Department of Computer Science University of Copenhagen, Denmark
5. Jones, M.T.: Anatomy of a cloud storage infrastructure / IBM developer works (November 30, 2010)
6. How Does Xen Work? (December 2009), http://www.xen.org/files/Marketing/HowDoesXenWork.pdf
7. Elhage, N.: Virtunoid: A KVM Guest ! Host privilege escalation exploit / Black Hat USA (2011)
8. Weinman, J.: Axiomatic Cloud Theory. Working Paper (July 29, 2011)
9. Chan, W.K., Mei, L., Zhang, Z.: Modeling and Testing of Cloud Applications. City University of Hong Kong and The University of Hong Kong (2009)
10. Li, Y., Boucelma, O.: A CPN Provenance Model of Workflow: Towards Diagnosis in the Cloud. Laboratoire des Sciences de l'Information et des Systémes, Domaine Universitaire de Saint-Jérôme
11. Ma, H., Schewe, K.D., Thalheim, B., Wang, Q.: A Formal Model for the Interoperability of Service Clouds (December 22, 2011)
12. Henzinger, T.A., Singh, A.V., Singh, V., Wies, T., Zufferey, D.: FlexPRICE: Flexible Provisioning of Resources in a Cloud Environment / IST Austria
13. Zegzhda, P.D., Zegzhda, D.P., Karetnikov, A.V.: Cloud systems. In: Virtual Security or Secure Virtualization? / Proc. of the Conference "RusCrypto" (2012)
14. Zegzhda, D.P., Karetnikov, A.V.: Cloud systems security. In: Problems and Prospect / (ISSN-2071-8217) Information Security Application #4 (2011)
15. Drouineaud, M., Luder, A., Sohr, K.: A Role based Access Control Model for Agent based Control Systems
16. Kalinin, M., Konoplev, A., Markov, Y.: Control of the security policies requirements in grid-systems. In: Proc. of the Conference Information Security of Russian Regions (ISRR 2011). St. Petersburg, Russia (2011)
17. Stepanova, T.: The relations between user behavior and outgoing network traffic for behaveiour anomaly detection. In: Proc. of the Conference Information Security of Russian Regions (ISRR 2011). St. Petersburg, Russia (2011)

Author Index

Achemlal, Mohammed 156
Avanesov, Tigran 130

Bakaev, Mihail 51
Birnbaum, Zachary 191

Cadenhead, Tyrone 36
Chechulin, Andrey 146, 270
Chen, Yu 286
Chevalier, Yannick 130
Coppolino, Luigi 171

Débar, Hervé 156, 203
Desharnais, Josée 114
Desnitsky, Vasily 146, 270
Dini, Gianluca 240
Dolgikh, Andrey 191

Gaber, Chrystel 156, 171
Gonzalez Granadillo, Gustavo 156
Grusho, Alexander 108
Grusho, Nick 108

Hutchison, Andrew 171

Jacob, Grégoire 156, 203
Jin, Xin 84
Johnson, Ryan 3

Kalinin, Maxim O. 299
Kantarcioglu, Murat 36
Kanyabwero, Erwanne P. 114
Kerrigan, Brendan 286
Khadilkar, Vaibhav 36
Komashinskiy, Dmitry 254
Konoplev, Artem S. 299
Korzhik, Valery 51
Kotenko, Igor 146, 254, 270
Krishnan, Ram 84

Livshits, Benjamin 1

Martinelli, Fabio 22, 240
Matteucci, Ilaria 22

Mjølsnes, Stig F. 65
Moldovyan, Alexandr 77
Moldovyan, Nikolay 77
Morales-Luna, Guillermo 51
Morisset, Charles 22
Murmuria, Rahul 3
Mustapha, Yosra Ben 203

Nikolskiy, Alexey V. 309
Novikova, Evgenia 77
Nykodym, Tomas 191

Oleshchuk, Vladimir 97

Prieto, Elsa 171

Rangwala, Huzefa 226
Rieke, Roland 171, 181
Rusinowitch, Michaël 130

Sandhu, Ravi 84
Saracino, Andrea 240
Schütte, Julian 181
Sgandurra, Daniele 240
Skormin, Victor 191
Stavrou, Angelos 3, 226
Stepanova, Tatiana V. 218

Tawbi, Nadia 114
Thuraisingham, Bhavani 36
Timonina, Elena 108
Tokhtabayev, Arnur 226
Tsay, Joe-Kai 65
Turuani, Mathieu 130

Wang, Zhaohui 3
Winkelvos, Timo 181

Yakovlev, Victor 51
Yavvari, Chaitanya 226

Zegzhda, Dmitry P. 218, 299, 309
Zegzhda, Peter D. 299, 309