International Symposium on Engineering Secure Software and Systems

February 27 - March 1, 2013
Paris (Rocquencourt), France

ESSoS Doctoral Symposium 2013

ESSoS Doctoral Symposium 2013

Maritta Heisel[1] and Eda Marchetti[2]

[1] paluno - The Ruhr Institute for Software Technology
University Duisburg-Essen,
Duisburg, Germany
maritta.heisel@uni-duisburg-essen.de
[2] Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR
via G. Moruzzi 1,
56124, Pisa, Italy
eda.marhetti@isti.cnr.it

Preface

These proceedings contain the accepted contributions of the second Doctoral Symposium of the International Symposium on Engineering Secure Software and Systems (ESSoS). The ESSoS Doctoral Symposium 2013 was held in Paris (Rocquencourt), France on Wednesday, February 27, 2013 in conjunction with the ESSoS 2013 Symposium.

Following the aim of the ESSoS-DS past edition, the scope of the current event was focused on providing PhD students an opportunity to discuss their research in Engineering Secure Software and Systems in an international forum, and with a panel of well-known experts in the field. In a welcoming and informal atmosphere, students could discuss the goals already achieved or planned, the research challenges they are interested in, the projects they are working on, the facilities they are developing, and the problems they want to solve in their doctoral work. During the Doctoral Symposium students received feedback from senior researchers, industrial partners and experts. The symposium was also a good opportunity for meeting and sharing experiences with other PhD students, who are addressing similar topics or are at a similar stage in their doctoral work.

All papers contained in these proceedings have been peer-reviewed by 3–4 programme committee members. They cover a wide spectrum of topics.

Francisco Moyano, Carmen Fernandez-Gago and Javier Lopez treat the topic of trust and reputation, which plays an important role in security-critical applications.

Naod Duga Jebessa, Guido van't Noordende, and Cees de Laat discuss how to construct secure virtual machines from declarative descriptions.

Jonathan Woodruff, Simon W. Moore, and Robert N. M. Watson propose a capability model to support memory segmenation, which enhances security.

Jan Stijohann and Jorge Cuellar aim at a systematic generation of security tests. They propose a two-step method, bridging the gap between risk analysis and security testing.

Christian Mainka, Vladislav Mlandeno, Juray Somorovsky, and Jörg Schwenk introduce a penetration test tool for XML-bse Web Services, which serves to secure service-oriented architectures.

Harsha K. Kalutarage, Siraj A. Shaikh, Qin Zhou and Anne E. James discuss a Bayesian approach to monitor nodes in computer networks for slow suspicious activities.

Tong Li, John Mylopoulos and Fabio Massacci consider socio-technical systems (STS), consisting of people, software, and hardware. They aim at developing a comprehensive framework for designing secure STS.

Alexander van den Berghe, Riccardo Scandariato and Wouter Joosen describe a procedure for a systematic literature review in the area of secure software design.

Katsiaryna Labunets and Fabio Massacci propose to conduct a series of empirical studies to aid practitioners in selecting an appropriate security requirements engineering method and to support method designers in improving their methods.

The diversity of the considered topics and applied techniques of the thesis research contained in these proceedings shows that the area of Engineering Secure Software and Systems is a lively one, offering challenging research opportunities.

Duisburg and Pisa, February 2013
Maritta Heisel and Eda Marchetti

Acknowledgments

We would like to thank the organizers of the ESSoS Symposium 2013 for the opportunity to hold the Doctoral Symposium in conjunction with the conference. Thanks also to the NESSoS Network of Excellence on Engineering Secure Future Internet Software Services and Systems for coming up with the idea of the Doctoral Symposium and for continuous support. We are grateful to the members of the programme committee who spent some of their valuable time for giving feedback to PhD candidates. Finally, we thank all the contributors for submitting their work to this Doctoral Symposium, making fruitful discussions possible.

Program Committee

Benoit Baudry	IRISA INRIA, Rennes
Ruth Breu	University Innsbruck, Innsbruck,
Jorge Cuellar	Siemens AG, München
Sandro Etalle	Technical University of Eindhoven, Eindhoven
Wouter Joosen	DistriNet Research Group KU, Leuven
Nora Koch	Ludwig-Maximilians-Universität München, München
Yves Le Traon	University of Luxembourg, Luxembourg
Javier Lopez	Universidad de Malaga
Fabio Martinelli	IIT-CNR, Pisa
Fabio Massacci	Università di Trento, Trento
Aljosa Pasic	Atos Origin, Madrid
Joachim Posegga	Passau University, Passau
Michaël Rusinowitch	LORIA-INRIA, Villers les Nancy
Christoph Sprenger	ETH Zurich
Ketil Stolen	SINTEF
Yijun Yu	Open University, Milton Keynes

Author Index

Table of Contents

A Trust and Reputation Framework *

Francisco Moyano

Department of Computer Science, University of Malaga, 29071, Málaga, Spain
moyano@lcc.uma.es

Abstract. The Future Internet is posing new security challenges as their scenarios are bringing together a huge amount of stakeholders and devices that must interact under unforeseeable conditions. In addition, in these scenarios we cannot expect entities to know each other beforehand, and therefore, they must be involved in risky and uncertain collaborations. In order to minimize threats and security breaches, it is required that a well-informed decision-making process is in place, and it is here where trust and reputation can play a crucial role. Unfortunately, services and applications developers are often unarmed to address trust and reputation requirements in these scenarios. To overcome this limitation, we propose a trust and reputation framework that allows developers to create trust- and reputation-aware applications.

Supervisors: Carmen Fernandez-Gago and Javier Lopez

1 Introduction: Problem and Motivation

Future Internet (FI) scenarios bring together multiple entities, namely stakeholders and devices, that need to collaborate in order to reach their goals. Should these entities knew each other beforehand, upfront mechanisms could be in place at design-time in order to ensure that these collaborations have a successful ending for all parties. However, this cannot be assumed. Therefore, it is required to guarantee a successful ending even under risky and uncertain conditions, which generally involves making good decisions. These conditions present a breeding ground for trust.

Even when the concept of trust is not standardized, it is agreed that it can be a valuable tool to leverage decision-making processes. The concept and implications of trust are embodied in trust models, which define the rules to process trust in an automatic or semi-automatic way within a computational setting. For the last twenty years, many models have been proposed, each one targeting different contexts and purposes, and with their own particularities.

*The research leading to these results have received funding from the European Community's Seventh Framework Programme FP7/2007-2013 as part of the Network of Excellence NESSoS (www.nessos-project.eu) under grant agreement number 256980. The first author is funded by the Spanish Ministry of Education through the National F.P.U. Program.

One issue with trust models is that they are usually built on top of an existing application in an ad-hoc manner in order to match the specific needs of the application and its environment, limiting the models' re-usability. Furthermore, most models do not distinguish explicitly between trust and reputation, nor do they provide guidelines to combine these notions to yield more solid results.

We believe that this approach is not adequate and that developers should be provided with some mechanisms to systematically incorporate trust and reputation models into their services and applications.

The rest of the paper is organized as follows. Section 2 describes the goals that we are pursuing. In Section 3 we explain the research methodology that is being followed and discuss the work that has been carried out up to now. Finally, the conclusions and some lines for future research are presented in Section 4.

2 Aims and Goals

Our main goal is the specification, design and implementation of a development framework that allows developers to implement trust- and reputation-aware applications. The framework must expose an Application Programming Interface (API) in order to make its functionalities accessible, and it must also provide hot spots where trust models can be customized to fit the application needs.

An important sub-goal that is derived from the main expected contributions is the provision of insight into trust and reputation. It is often the case that these concepts are considered as being synonyms or are used interchangeably, however they are quite different notions that need to be considered separately. Building a development framework requires performing a domain analysis in the framework targeted area, in this case, trust and reputation. Not only can this domain analysis shed light on concepts such as trust or reputation, but also on the trust models internal workings.

The main expected contribution of this research to the field of Engineering Secure Software and Systems is two-fold: on the one hand, by providing developers with a tool like a trust and reputation framework, we foster thinking over trust and reputation requirements from the very beginning. On the other hand, as applications are developed by using the framework, trust and reputation models are naturally incorporated within the application itself, and not as patches added after-the-fact, as it is the standard nowadays. Thus, trust models can use all the information available to the application in a more efficient way.

3 Research Methodology

This section summarizes the research methodology that is being followed. It is divided into six phases, each one further elaborated in its own section. For each phase, we state whether it is completed or further work needs to be done, and we also outline their main findings and results.

3.1 Phase 1: Literature Review

Surveys, such as the one by Jøsang, Ismail and Boyd [5] or the one by Ruohomaa and Kutvonen [11] are the best starting point to obtain a solid knowledge of the work carried out in trust and reputation over a period of time, and they constitute the main source for the next phase: the domain analysis. Other interesting contributions include those that provide assistance to developers with creating trust and reputation implementations. In this direction, we conducted research on architectural styles [12], frameworks [7] [2] and middlewares [6] [4] where trust and reputation are the core concept.

Some drawn conclusions are that most works do not provide enough margin of customization and lack of a framework-oriented approach. In addition, no existing contributions differentiate between the notions of trust and reputation, as they tend to focus on just one of them, usually reputation.

Even though this task was already finished, it is required to continuously check out new interesting papers.

3.2 Phase 2: Domain Analysis

A domain analysis is of paramount importance when building a development framework [3], and for this analysis to be complete, it may be required to look up new literature that helps to fill some gaps that may have arisen.

The main contribution is a conceptual framework that gathers and relates the most important concepts in trust and reputation models. This framework is represented in the form of UML diagrams, like the one depicted in Figure 1. As explained in an earlier contribution [8], this conceptual framework also serves as a comparison framework under which different trust and reputation models can be compared.

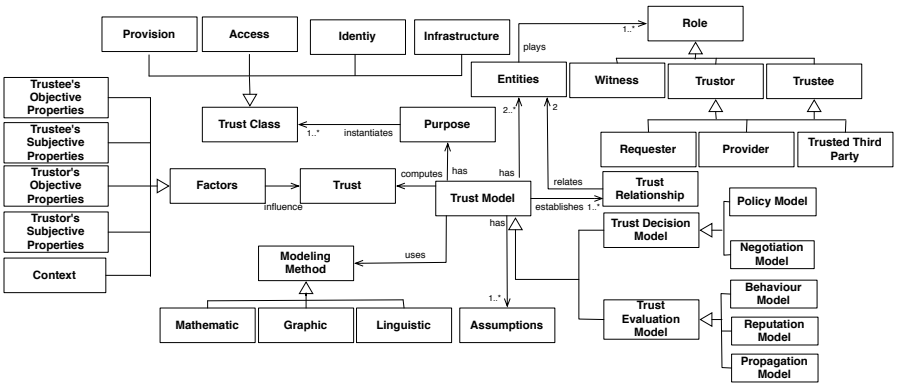


Fig. 1: Common Concepts for Trust Models

Also, in our earlier work [8], we classified trust models into two types: decision models, tightly related to the authorization problem, and evaluation models, where the evaluation of trust according to several influencing factors is the most important consideration.

Even though this phase is also completed, the analysis should be refined as new relevant papers arise.

3.3 Phase 3: Requirements Elicitation

The previous phase conducted an exhaustive analysis on trust and reputation. This analysis assisted in determining the requirements that a trust and reputation framework must fulfil. Since accommodating all possible trust models in a single framework may be a daunting task, we decided to focus on evaluation models. A list of requirements can be found in an earlier work [10].

Even though this phase is finalized, some new requirements may arise as a consequence of new relevant literature or due to the architecture and design phases. One of our findings is that evaluation models are centred around the notion of trust metric. Trust metric uses a computation engine to yield a trust or reputation value given a set of variables. The difference between many evaluation models stems from (i) the variables used in the computation and (ii) the computation engine used to aggregate these variables into a simple value or a tuple of values. Therefore, one of the most important requirements for a trust and reputation framework is to allow developers to define their own metrics. Time and uncertainty are two important factors as well, and developers should be allowed to include them. The former may be used to consider freshness in trust values, whereas the latter refers to how reliable a trust value is.

3.4 Phase 4: Architecture

This phase consists of sketching the high-level software structure that supports the requirements elicited in the previous phase. A half-way technical and conceptual architecture was discussed in earlier works [10] [9]. A recent work[1] provides greater insight into the technical details of a possible architecture, and even guidelines are given for implementation of some of the components and their communication mechanisms.

At the architectural level, building a trust and reputation framework requires planning in two fronts. On the one hand, we need to carefully design an easy yet flexible API that allows connecting any application to a trust server. On the other hand, the framework must provide enough hot spots to support the customization of the trust server behaviour at runtime in order to accommodate new trust and reputation models.

The type of application that we want to build by using the framework determines the design of the aforementioned factors: API and hot spots. In this

[1] We cannot provide the reference as the work is currently under review.

sense, we think of two types of applications that follow two different architectural styles: client-server applications and peer-to-peer applications.

The first one requires the developer to define the interactions between an application server and a trust server, and the trust server holds information about the whole system. In the second approach, each peer holds an instance of the trust server, which holds only partial information about the whole system.

The architecture proposed in our recent work[1] was originally designed to support the client-server approach, even though we think it can be tailored in order to support the peer-to-peer architectural style.

3.5 Phase 5: Design and Implementation

This phase elaborates on the architecture in order to refine the components into sub-components and modules. Inner data structures are also detailed and the database schemas and tables are fully specified. This refinement goes on until the implementation of each module is made easy. This phase remains unfinished.

3.6 Phase 6: Validation

The last phase consists of validating the framework implementation by developing a trust-aware application in the scope of e-Health and/or SmartGrid, which have been identified as the two main NESSoS[2] scenarios [1].

It is likely that we observe certain deficiencies and limitations of the framework in a real application. Actually, any framework requires iterations in order to be able to accommodate a wide range of applications. Therefore, the output of this phase could help to improve the architecture and design of the framework.

4 Conclusions and Future Work

New Future Internet applications will need support from trust and reputation services for their successful adoption. Yet these services have been laid aside and are very often considered once an application is already deployed and running. At that moment, adding trust and reputation features may be hard, and may lead to poor and, above all, barely reusable solutions.

We propose a trust framework that assists developers in the task of creating services and applications that need trust and reputation models. Examples of such applications are those proposed in the NESSoS project, and validation is to be done in their scope.

As future work, we are planning to research on how reconfiguration mechanisms can leverage trust models during the service or application lifetime. The trend in Software Engineering is towards adapting the software at runtime to new requirements or new environmental conditions, changing the architecture itself without the need for re-implementation. We would like to obtain insight into how the trust framework could exploit advances in this direction in order to support self-adapting trust models.

[2] www.nessos-project.eu

References

1. Selection and Documentation of the Two Major Application Case Studies. NESSoS Deliverable 11.2, October 2011.
2. Vinny Cahill, Elizabeth Gray, Jean-Marc Seigneur, Christian D. Jensen, Yong Chen, Brian Shand, Nathan Dimmock, Andy Twigg, Jean Bacon, Colin English, Waleed Wagealla, Sotirios Terzis, Paddy Nixon, Giovanna di Marzo Serugendo, Ciaran Bryce, Marco Carbone, Karl Krukow, and Mogens Nielsen. Using Trust for Secure Collaboration in Uncertain Environments. *IEEE Pervasive Computing*, 2(3):52–61, July 2003.
3. Mohamed E.Fayad, Douglas C.Schmidt, and Ralph E.Johnson. *Building Application Frameworks: Object-Oriented Foundations of Framework Design.* Wiley, Septembre 1999.
4. Chern Har Yew. *Architecture Supporting Computational Trust Formation.* PhD thesis, University of Western Ontario, London, Ontario, 2011.
5. Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, March 2007.
6. Rolf Kiefhaber, Florian Siefert, Gerrit Anders, Theo Ungerer, and Wolfgang Reif. The Trust-Enabling Middleware: Introduction and Application. Technical Report 2011-10, Universitätsbibliothek der Universität Augsburg, Universitätsstr. 22, 86159 Augsburg, 2011. http://opus.bibliothek.uni-augsburg.de/volltexte/2011/1733/.
7. Adam J. Lee, Marianne Winslett, and Kenneth J. Perano. TrustBuilder2: A Reconfigurable Framework for Trust Negotiation. In Elena Ferrari, Ninghui Li, Elisa Bertino, and YÃ$\frac{1}{4}$cel Karabulut, editors, *IFIPTM*, volume 300 of *IFIP Conference Proceedings*, pages 176–195. Springer, 2009.
8. Francisco Moyano, Carmen Fernandez-Gago, and Javier Lopez. A conceptual framework for trust models. In Simone Fischer-Hübner, Sokratis Katsikas, and Gerald Quirchmayr, editors, *9th International Conference on Trust, Privacy & Security in Digital Business (TrustBus 2012)*, volume 7449 of *Lectures Notes in Computer Science*, pages 93–104, Vienna, Sep 2012 2012. Springer Verlag, Springer Verlag.
9. Francisco Moyano, Carmen Fernandez-Gago, and Javier Lopez. Implementing trust and reputation systems: A framework for developers' usage. In *International Workshop on Quantitative Aspects in Security Assurance*, Pisa, Sep 2012 2012.
10. Francisco Moyano, Carmen Fernandez-Gago, and Javier Lopez. Building trust and reputation in: A development framework for trust models implementation. In *8th International Workshop on Security and Trust Management (STM 2012)*, Pisa, In Press.
11. Sini Ruohomaa and Lea Kutvonen. Trust management survey. In *Proceedings of the Third international conference on Trust Management*, iTrust'05, pages 77–92, Berlin, Heidelberg, 2005. Springer-Verlag.
12. Girish Suryanarayana, Mamadou H. Diallo, Justin R. Erenkrantz, and Richard N. Taylor. Architectural Support for Trust Models in Decentralized Applications. In *Proceeding of the 28th international conference*, pages 52–61, New York, New York, USA, 2006. ACM Press.

Towards Purpose-Driven Virtual Machines

Naod Duga Jebessa Guido van 't Noordende Cees de Laat

University of Amsterdam
Science Park 904, 1098XH Amsterdam, The Netherlands
{jebessa,noordende,delaat}@uva.nl

Abstract

Virtual machines (VMs) are often generic though they are meant to serve a specific purpose. The redundancy of generic VMs may incur costs in *security* (due to bigger attack surface and a larger trusted computing base) and *performance* (due to extra VM image size as well as overheads in CPU and memory). We are working on techniques to build minimal, application-specific and secure virtual machines from declarative descriptions. An application running in a VM has dependencies on other applications, libraries, kernel features and (virtual) hardware. We model such a dependency as a graph. The VM is treated as an optimized system built specifically to satisfy this dependency out of a set of interdependent components such as packages and kernels from OS distributions. Such distributions and installations based on them are inherently complex networks that could benefit from a formal, rigorous treatment in order to perform security and performance optimization. In this paper, we describe our motivation and vision for this project, outline our approach, briefly report on current status, discuss challenges and research problems that we intend to work on in the future.

1 Introduction

The virtual machine concept is one of the key enablers of clouds, and has found applications in mobility, security, fault tolerance, grids, testing, and development. VMs are often used to run a set of applications and as such can be thought of as having a specific purpose. Nonetheless, VMs are commonly instantiated from ready-made, generic images or are composed from off-the-shelf components like applications, libraries, operating systems, and device drivers meant for physical machines. For example, a VM image with a general purpose kernel with millions of lines of code (LOC) in its trusted compute base (TCB) can result in an unnecessarily large attack surface. There are vulnerable libraries, applications, protocol stacks and service configurations that could be omitted or otherwise optimized without breaking the functionality of the application in question.

VMs are prone to misconfiguration by users who have to work with generic images that are not optimized, either due to lack of expertise or simply because it is easier to just use an existing, general purpose image. This may incur costs in terms of *performance*, due to extra VM image size, overheads in CPU and memory usage, as well as *security*, due to bigger attack surface and a larger TCB. Moreover, users are faced with VM sprawl as images are created, deployed, changed, snapshotted, cloned and archived, making lifecycle management difficult. Many of these problems can be attributed to the black-box nature of VM images.

We are advocating declarative descriptions of virtual machines so we can build an optimal virtual machine on

In: M. Heisel, E. Marchetti (eds.): Proceedings of ESSoS-DS 2013, Paris, France, 27-Feb-2013, published at http://ceur-ws.org

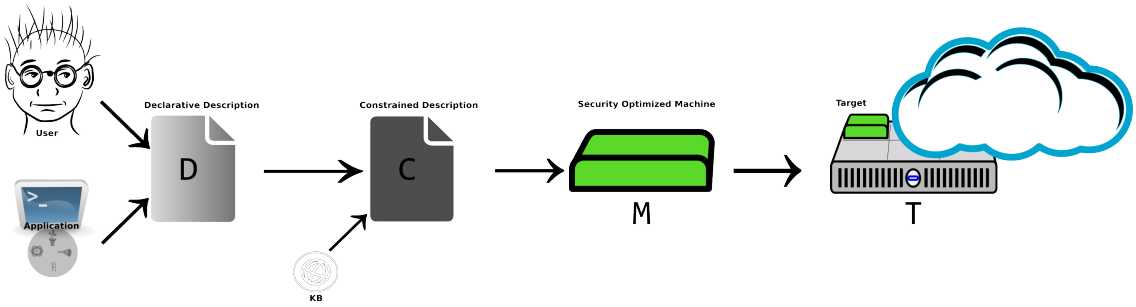


Figure 1: Outputs D, C, M & T after each of the 4 stages in the VM building pipeline

demand from existing components. Doing so should allow us to do fine grained and automated optimization. Moreover, it would be easier to reason over the security or other properties of the system using a semantically rich model, than over an unstructured virtual machine image.

2 Approach

We use graph models for OS distributions and virtual machines built from them. The purpose of the VM is declared using a domain-specific language (DSL [17, 23]). The declarative description is then used as an input to a pipeline used to build an optimized VM. The problem is partitioned into two major subproblems. The first subproblem, corresponding to the two stages in our four stage pipeline (see D and C in figure 1), focuses on descriptions and as such involves language and semantics related issues as well as ways to capture user/application requirements. The second subproblem is about translating the descriptions to concrete systems that meet constraints derived from the second stage. For brevity, we have shown the outputs of each stage while arrows represent intermediate blocks.

2.1 Operating Systems as Complex Networks

It is possible to model an operating system distribution as a graph of interdependent components. An operating system OS_i can have v versions, each comprising a set P of packages with a dependency graph G_p and a kernel whose features [4, 7] can be represented as a dependency graph G_k. It is worth noting that there is an implicit dependency between G_p and G_k as applications and libraries expect a kernel to interact with and because many distributions treat the kernel itself as a package.
Because we can annotate the graph model with external information about, say, security-related statistics on packages, we will be able to employ security-aware constraints in our optimization pipeline. This pipeline is part of a toolchain for building virtual machines that we are developing.
An ongoing work is to use ideas from network science and complexity [1, 2] to understand the structure and dynamics of OS distributions and installations [5]. As an example, we created an annotated semantic graph of `Debian Squeeze` that comes with a `Linux 2.6` kernel for `i386`. This graph has approximately 40 thousand nodes and 170 thousand edges. Figure 2 shows a visualization of the graph. One can imagine how challenging it is to manipulate this model and gain insight from it. However, we simplify the problem by focusing on the dependency graph of a small set of applications on a specific OS, as shown in figures 3 and 4.

2.2 Declarative Descriptions and Constraints

The VM is declaratively described in a domain-specific language (DSL). The description specifies minimal requirements from the VM such as set of packages, kernel features and system configurations for software, (virtual) hardware, networks, and so on. This description is then analyzed to fill in details and create a constraint for optimization, as shown in figure 1. The DSL should be expressive enough and should allow explicit declarations

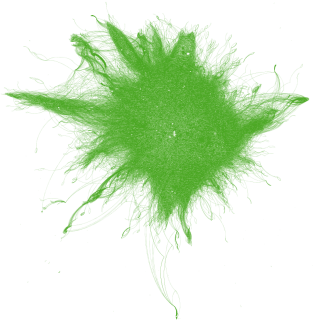


Figure 2: An OS as a complex network

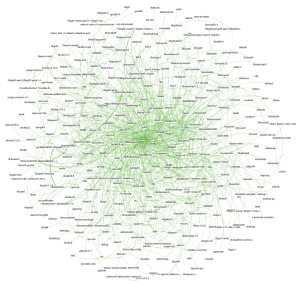


Figure 3: Minimized network for a specific application. The center node is `libc`, the C runtime library for Unix-like systems.

through conjunction, negation, and disjunction. For example, a description may have a list of requirements that must be satisfied (conjunction), unneeded functionality (negation), and alternatives for a certain feature (disjunction). Implicit requirements in the VM will be incorporated as default constraints upon analysis. For example, if the declarative description does not state what version of an OS to use, a default version may be selected. Declarative descriptions and constraints have several advantages. They can be versioned to allow rollback; can be based on existing templates; and can be reasoned about (say, to map a declaration/constraint to an already built VM).

2.3 Optimization and Reasoning About VM Security

We look at the VM as the minimal composition from a set of interdependent components in package dependency graphs G_{p_i} and kernel feature graphs G_{kj} that satisfies the constraint set forth by the declarative description. Dependency resolution is an NP-complete problem [3]. The fact that we add constraints might lead to declarations with constraints that are not satisfied. From the package dependency graph point of view (see figure 4 for a specific application[1]), the problem is finding an optimal installation that satisfies constraints pertaining to packages [8]. For the kernel, this translates to (de)selecting features based on constraints on the kernel, specifically device drivers, file systems and protocol stacks [4, 7].

Let us consider the case where there is a security advisory for a network exploitable vulnerability in package `X` version `a.b.c` for a specific OS. A query to the declarative description and the blueprint of the VM would show if we have that specific version in the virtual machine. This could be automated and can be an integral part of a simplified security lifecycle management pipeline.

A practical problem with this approach is that security advisories (like CVE/DSA[2]) are not semantically rich enough. We see promising research with practical significance in employing semantic technologies to describe, consume and process security-related information.

2.4 Building Virtual Machines

Having installed an optimized package set on top of a possibly optimized kernel, a VM image can be configured with cryptographic keys, users, passwords and the like along with extra dependencies like memory, processor, storage and network settings to a specific target hypervisor.

We are building a proof-of-concept setup to validate the ideas presented so far. There are at least four engineering challenges. First, there are quite a few hypervisors to target, each with a somewhat different view of a guest

[1] `http://fsl.fmrib.ox.ac.uk`

[2] `http://cve.mitre.org/`, `http://www.debian.org/security/`

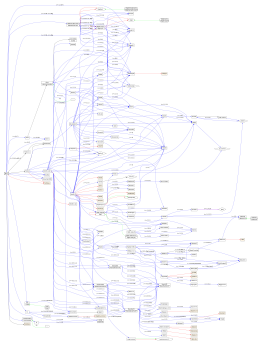


Figure 4: Partial dependency graph for a specific application

VM. To this end, we plan to have a modular and extensible VM building pipeline. The design of the hypervisor might also constrain how we optimize the VM and it is worth considering hypervisor-VM optimization in this regard, e.g. with respect to drivers.
Second, there are a range of VM image formats and containers to target. The *qcow2* format used by KVM/QEMU, for instance, is quite different from formats like Xen raw disk, AMI (Amazon, with separate kernel and ramdisk), Microsoft Hyper-V's *vhd*, VirtualBox *vdi*, and *vmdk* (VMware). However, a modular design should allow us to build a VM and target it to different formats. This is possible because most formats are more or less indifferent to how we optimize the internal subsystems of the VM and because there are readily available conversion tools.
Third, there are a plethora of operating systems that we can build virtual machines from. Modeling each OS as an annotated dependency graph is a difficult endeavor. Hence, we plan to work on a set of representative OS distributions. Even though our approach is ideal for the Unix philosophy and package-based source/binary OS distributions, it should be possible to extend the idea to other platforms, as the formalism we are developing is fairly generic.
Last but not least, there is the issue of efficient VM building. One might consider having a ready made 'base' VM setup for commonly used virtual machines and configure a candidate based on the constraints set forth by the declarative description. This approach may be useful when the time to build and deploy is of significant importance, as building a custom kernel takes several minutes and package installation and configuration takes significant time as well.

3 Research Challenges and Future Work

1. *How do we manage complexity and heterogeneity?*: One challenge is coming up with a fairly generic abstraction that can be used to model a VM based on any one of a range of OS distributions; built for one or more hypervisors and processor architectures; and can be targeted at one or more cloud stacks. We believe that dependency graphs and complex networks are a viable foundation to start with, together with an information model for each component from which the VM is composed from. In such an approach, we treat the VM as a layered stack of dependencies in hardware, hypervisor, kernel features, libraries, applications and configurations. Such a graph model allows us to separate theoretical problems from implementation issues. The model would benefit from existing research in complex networks [1], graph theory [2, 12, 21], boolean satisfiability and dependency resolution [3, 19, 13, 14]. We could leverage existing tools and demonstrate our ideas through a proof of concept pipeline instead of a complete implementation that supports all major operating systems and hypervisors.

2. *How do we describe the specific purpose of a VM?*: People often use natural languages to describe what a specific machine is for, be it physical or virtual, even though the machine is, in principle, general-purpose.

The design of a declarative language for VM description calls for a tradeoff between simplicity and expressiveness. The first challenge arises from the fact that a user cannot be expected to explicitly declare everything. So the language [20, 23] should have constructs to support implicit declarations. Second, the expressive power of the language dictates much of what is done after processing a description. For example, a very fine-grained description (e.g. with conjunctive normal form constraints) may be difficult to satisfy, due to a limited search space and making the optimization problem at hand difficult (as in too many variables and constraints) and time-consuming (translating the solution to a concrete VM). A coarse-grained description, on the other hand, may introduce indeterminism, sub-optimal or too-optimized solutions, and an increased search space.

3. *How do we fill the semantic gap in security?*: To our surprise, much of the available security-related information (e.g. from CVE or specific vendors) is not structured enough to be used by automated tools. In our particular case, for example, it is not easy to gather vulnerability information so as to measure or evaluate the security of a VM instance due the ambiguity, incompleteness and a non-standard vocabulary, to mention a few, of security advisories. Along with our work in measuring security of a VM and on attack surface evaluation metrics, we plan to suggest a semantically rich, machine-readable format to describe vulnerabilities and develop a logic formalism for attack scenarios, specially for use by OS distributions like Debian.

4. *What is this all for?*: In cloud parlance, our idea is conceptually equivalent to offering a PaaS on demand, targeted at an existing IaaS. A good example might be creating a VM cluster for a DNA processing application that expects a certain runtime environment, including an OS installation and all required dependencies. Hence, our project essentially maps high-level application requirements to low-level infrastructure details. Moreover, some application scenarios are security and privacy sensitive. Hence, we are advocating a disciplined way of building VMs with the hypothesis that a pipeline that translates declarative descriptions to concrete VMs would allow for 'white-box machines' (whose internal manifest is clearly known), hence allowing us to reason about certain attributes of such a VM like its trusted compute base (TCB) size, the size of its attack surface, the possibility of exploitation given an attack scenario, or trustworthiness in general. While security is the main goal of our work, the approach could have advantages in smaller VMs that could perform well [18] and are easy to maintain, allowing users and frontend tools to automatically build virtual machines. As future work, we plan to build demos and explore use cases and usability as the feedback from such endeavors will allow us to refine our approach to the problem.

5. *Validation and Evaluation Techniques*: We are working on the assumption that optimized, purpose-driven VMs are likely to be more secure and could possibly perform better. In addition, our approach of employing a declarative VM description language might improve usability, flexibility and maintainability. All the above hypotheses need to be tested. The security of a VM is not straightforward to quantify due to the absence of an evaluation metric. However, having the knowledge of the VM internals should allow us to propose novel metrics that we can use to compare the security of optimized VMs with their generic counterparts. Performance of the VM building pipeline and the built VMs, on the other hand, can readily be evaluated in terms of time taken to build a VM (compared to manual building and off-the-shelf deployment of an existing VM image); the CPU and memory usage while the VM is running; and VM image size (which is important during migration, for example). Usability, flexibility and maintainability could be evaluated through use cases that make use of our tools, albeit subjectively.

4 Related Work

There are quite a few projects that aim at building VMs [11], image manipulation[3], configuration tools & APIs[4], and interaction with hypervisors[5], the majority of which are open source. We are aware of at least half a dozen solutions focusing on VM building, with varying maturity and OS support. Our work differs in two major ways and is, in essence, orthogonal to most. First, building a VM is only part of what we do and as such it can be considered as an implementation issue. Second, we are introducing a purpose-driven (declarative) VM-building paradigm wherein we do fine-grained optimization with implications in security, performance and flexibility, as opposed to straightforward installation as is often done by existing solutions[6].

[3] `http://libguestfs.org/`
[4] `http://augeas.net/`, `http://cfengine.com/`
[5] `http://libvirt.org/`
[6] `http://wiki.debian.org/VMBuilder`, `http://virt-manager.org/`

Few authors have studied operating system distributions as complex networks [1] and we believe there is more to be done, specially in the context of security, OS complexity and virtual machines. The authors in [3, 5] have done detailed studies on formalisms and tools to manage complexity of package-based OS distributions showing that they exhibit the small world property [2] of many networks. We hope to work towards a graph theoretic attack surface metric [10] that takes into consideration semantic metadata about components and the composition thereof, of a virtual machine. This would allow us to evaluate the security advantages of minimal and application-specific VMs.
In the domain of software engineering, researchers have studied automatic generation [15, 16, 22], predictable assembly [6], variability and feature models [4, 7, 9] as applied to OS kernels.

5 Conclusion

For many users, a virtual machine is a black box that is difficult to reason about, trust, configure and maintain. While this is most certainly true for inexperienced users, expert users also find themselves working with pre-configured or generic VM images, which is anything but transparent. Furthermore, automated infrastructure provisioning tools are increasingly using virtual machine technology to serve computational needs by deploying VMs on demand, often from pre-built images. We believe that there is a missing link between the *purpose of a VM* on one hand and *the way a VM is built* on the other. We need a generic and disciplined process to build our virtual machines.
Our work fits in the wider problem of security and privacy in clouds. One aspect is having trustworthy VM installations that can be (remotely) attested. The user might want to just declare and expect to get a fairly secure VM while the cloud provider might expect to have some guarantee that the VM is 'safe' to deploy. Depending on the use case, the declarative description and intermediate steps in our pipeline could be used or extended to incorporate mutual trust, where both parties can verify relevant properties of the virtual machine. Another aspect is the issue of privacy in clouds which we believe can benefit from a trustworthy execution environment. If a VM can be attacked and owned, it will be a breach of the necessary condition of confidentiality of the data it is supposed to process (as part of its *purpose*), secret keys it has stored, etc.
In this paper, we have described the idea of a *purpose-driven virtual machine*. Our motivation is the observation that a VM is often meant to serve a specific purpose and the fact that generic VMs have redundant components that may be omitted, giving advantages in security and performance. Using declarative descriptions of application requirements, we are able to generate virtual machines on demand out of component ecosystems (i.e. OS distributions) modeled as complex networks. We have discussed our approach and outlined some theoretical and practical problems that we intend to work on in the future.

Acknowledgment: This research is supported by the Dutch national research program COMMIT (*http://www.commit-nl.nl*). We would like to thank the anonymous reviewers whose comments helped improve the paper.

References

[1] M. Newman, A-L. Barabasi, and D.J. Watts. 2006. The Structure and Dynamics of Networks: (Princeton Studies in Complexity). Princeton University Press, Princeton, NJ, USA.

[2] J. Kleinberg. 2000. The small-world phenomenon: an algorithm perspective. In Proceedings of the thirty-second annual ACM symposium on Theory of computing (STOC '00). ACM, New York, NY, USA, 163-170.

[3] F. Mancinelli, J. Boender, R. di Cosmo, J. Vouillon, B. Durak, X. Leroy, and R. Treinen. 2006. Managing the Complexity of Large Free and Open Source Package-Based Software Distributions. In Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering (ASE '06). IEEE Computer Society, Washington, DC, USA.

[4] T. Berger, S. She, R. Lotufo, A. Wąsowski, and K. Czarnecki. 2010. Variability modeling in the real: a perspective from the operating systems domain. In Proceedings of the IEEE/ACM international conference on Automated software engineering (ASE '10). ACM, New York, NY, USA, 73-82.

[5] P. Abate, R. di Cosmo, J. Boender, and S. Zacchiroli. 2009. Strong dependencies between software components. In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09). IEEE Computer Society, Washington, DC, USA, 89-99.

[6] S.A. Hissam, G.A. Moreno, J.A. Stafford, and K.C. Wallnau. 2002. Packaging Predictable Assembly. In Proceedings of the IFIP/ACM Working Conference on Component Deployment (CD '02), J.M. Bishop (Ed.). Springer-Verlag, London, UK, UK, 108-124.

[7] R. di Cosmo and S. Zacchiroli. 2010. Feature diagrams as package dependencies. In Proceedings of the 14th international conference on Software product lines: going beyond (SPLC'10), J. Bosch and J. Lee (Eds.). Springer-Verlag, Berlin, Heidelberg, 476-480.

[8] P. Abate, R. di Cosmo, R. Treinen, and S. Zacchiroli. 2013. A modular package manager architecture. Inf. Softw. Technol. 55, 2 (February 2013), 459-474.

[9] L. Passos, K. Czarnecki, and A. Wąsowski. 2012. Towards a catalog of variability evolution patterns: the Linux kernel case. In Proceedings of the 4th International Workshop on Feature-Oriented Software Development (FOSD '12), Ina Schaefer and Thomas Thm (Eds.). ACM, New York, NY, USA, 62-69.

[10] P.K. Manadhata and J.M. Wing. 2011. An Attack Surface Metric. IEEE Trans. Softw. Eng. 37, 3 (May 2011), 371-386.

[11] I. Krsul, A. Ganguly, J. Zhang, J.A. B. Fortes, and R.J. Figueiredo. 2004. VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In Proceedings of the 2004 ACM/IEEE conference on Supercomputing (SC '04). IEEE Computer Society, Washington, DC, USA, 7-.

[12] M. Newman. 2010. Networks: An Introduction. Oxford University Press, Inc., New York, NY, USA.

[13] G. Jenson, J. Dietrich, and H.W. Guesgen. 2010. An empirical study of the component dependency resolution search space. In Proceedings of the 13th international conference on Component-Based Software Engineering (CBSE'10), L. Grunske, R. Reussner, and F. Plasil (Eds.). Springer-Verlag, Berlin, Heidelberg, 182-199.

[14] L. Bordeaux, Y. Hamadi, and L. Zhang. 2006. Propositional Satisfiability and Constraint Programming: A comparative survey. ACM Comput. Surv. 38, 4, Article 12 (December 2006).

[15] L. Guthier, S. Yoo, and A. Jerraya. 2001. Automatic generation and targeting of application specific operating systems and embedded systems software. In Proceedings of the conference on Design, automation and test in Europe (DATE '01), W. Nebel and A. Jerraya (Eds.). IEEE Press, Piscataway, NJ, USA, 679-685.

[16] A. Sarmento, L. Kriaa, A. Grasset, M-W. Youssef, A. Bouchhima, F. Rousseau, W. Cesario, and A.A. Jerraya. 2005. Service dependency graph: an efficient model for hardware/software interfaces modeling and generation for SoC design. In Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES+ISSS '05). ACM, New York, NY, USA, 261-266.

[17] A. van Deursen, P. Klint, and J. Visser. 2000. Domain-specific languages: an annotated bibliography. SIGPLAN Not. 35, 6 (June 2000), 26-36.

[18] N.D. Jebessa, G. van 't Noordende, C. de Laat. 2012. Optimizing Security for Virtual Machine Applications. In HPDC 2012. The 21st International ACM Symposium on High-Performance Parallel and Distributed Computing (Poster Abstract). (July 2012)

[19] T. Zimmermann, and N. Nagappan. 2007. Predicting subsystem failures using dependency graph complexities. In Software Reliability 2007. ISSRE'07. The 18th IEEE International Symposium on, pp. 227-236. IEEE, 2007.

[20] R.J. Stainton. 1996. Philosophical perspectives on language. Peterborough, Ont., Broadview Press.

[21] S.E. Schaeffer. 2007. Survey: Graph clustering. Comput. Sci. Rev. 1, 1 (August 2007), 27-64.

[22] K. Czarnecki and U.W. Eisenecker. 2000. Generative Programming: Methods, Tools, and Applications. ACM Press/Addison-Wesley Publ. Co., New York, NY, USA.

[23] M. Mernik, J. Heering, and A.M. Sloane. 2005. When and how to develop domain-specific languages. ACM Comput. Surv. 37, 4 (December 2005), 316-344.

Memory Segmentation to Support Secure Applications

Student: Jonathan Woodruff
Supervisor: Simon W. Moore
Project Lead: Robert N. M. Watson

University of Cambridge, Computer Laboratory

Abstract.

Current CPU architectures provide only weak support for software segmentation, a key underpinning for software security techniques such as sandboxing, managed languages, and static analysis. Because hardware memory segmentation is relevant mainly in the program abstraction its support has been deemphasized in modern operating systems, yet modern hardware requires operating system support to use its segmentation features. This paper argues that by implementing a *capability* model, it is possible to safely support creation, distribution and use of segments purely in user space. Hardware support for user-mode segmentation would enable efficient sandboxing within processes, enforcement of compiler structure, managed languages, and formal verification of machine code. We present a working prototype of such a system implemented on an FPGA and running FreeBSD.

1 Introduction

Sandboxing, managed language runtimes, bounds checking, and static analysis are software vulnerability prevention and mitigation techniques that rely on the idea of memory segmentation, that is, that memory should be addressed as independent objects with sizes and properties rather than as a single contiguous space. However modern software has gone to great lengths to implement segmentation using hardware primitives not intended for the purpose. Examples include sandboxing, which has used process separation [18] using the paged memory translation lookaside buffer, or TLB [20], and managed languages, which use a blend of static analysis and general purpose instructions for bounds checking [15].

A dominant instruction set architecture, IA-32, actually supports segmentation in user space, however it depends on the operating system for protection and manipulation [2]. We observe that fine-grained segmentation features are primarily useful in the program abstraction, yet available hardware segmentation models have required management from the operating system [19]. This mismatch, and the fact that RISC architectures did not arrive at a consensus model for segments, meant that hardware segmentation has not become a useful feature to the modern software stack.

We have developed a RISC segment model which maintains the single-cycle instruction principle and a load store architecture. Our segment model allows creation and distribution of segments along with protection domain crossing entirely in user space. Our model is intended to support a full *capability* model [13] within a RISC processor. This capability segment model can efficiently support sandboxing [10], managed language runtimes, and static analysis. We have implemented our model as an extension to the 64-bit MIPS instruction set and have a working prototype on an FPGA which runs FreeBSD. We are beginning to evaluate our approach through a set of application use cases that include sandboxing and language runtimes and this will require the development of software models and an evaluation methodology.

2 Background and Motivation

2.1 History of Segmentation

In the 1950s and 1960s, the expanding size of computer programs made obvious the need for memory protection and virtualization [5, 17]. Programmers of the time identified segments as a direct solution for their require-

ments, allowing them to protect and relocate program modules [6]. However a more forward-looking proposal in 1961 suggested that if programs were willing to sacrifice precise control over protection and virtualization to use regularly sized pages, an operating system could efficiently allow multiple programs to run on the same machine [8]. Large-scale computers generally moved to paging to support operating systems with multiprogramming [16]. Nevertheless, computers that are expected to run a single program often support segmentation, for example the Intel 286 [2] and the modern ARM Cortex R cores [9]. As computer designers preferred paging to segmentation, programmers eventually found that strict high-level languages could enforce memory protection to some degree [4] and that managed languages could actually verify memory references at run-time [1]. Thus hardware segmentation fell out of use for large computer programs.

Intel's IA-32 instruction set demonstrates this historical pattern. The Intel 286 was the first Intel x86 processor to support memory protection [2], but because it was not expected to be used primarily for multiprogramming, a segmentation model was devised that might support protection for program objects. However segments were found to be awkward to use since segment descriptors were manipulated in kernel mode, a higher privilege level than that of the program which was creating and using the objects. The next generation of x86 processor, the 386, supported paging in addition to segmentation to enable standard multiprocess operating systems. Eventually the segmentation system was rarely used and was mostly dropped in the transition to x86-64 [3].

2.2 Motivation for Renewed Segment Support: Security

While strict programming languages and managed runtimes have been sufficient to ensure a level of working correctness, both have shown cracks when facing intentional exploitation [15]. These flaws were often due to compilers and runtimes being programmed in unsafe languages themselves. Recurrent and escalating problems with security due to the ever more connected global network of computers have shown that a working correctness is not sufficient for security conscious programs.

Maintaining security in the face of a global collection of capable adversaries requires very strong correctness of security mechanisms. While securing individual client computers does not automatically make a network secure, many network intrusions begin with a violation of a client's memory protection. A comprehensive hardware segmentation system would make direct, automatic and continual protection of objects in memory available to compilers and runtimes to improve performance and simplify enforcement.

2.3 The Capability System Model: A Guide for a Comprehensive Segment Model

In 1966 Computer Science began to clearly discuss a cohesive, decentralized system for program correctness and security [7]. The result was the development of *capability* theory. A capability is an unforgeable token of authority which can be used by a program component and which can be manipulated according to rules and delegated to another program component. Computer scientists spent much effort toward proving that capability systems could be correct and safe and even constructed several hardware capability systems [13] [14], including commercial systems from IBM [12] and Intel [11]. Many of these hardware capability systems were the pinnacle of program-centered, segment-based computers as their memory protection systems often did not depend on a central authority for inter-domain transactions, but on a set of hardware enforced rules that allowed safe, direct sharing between program components. These hardware capability systems were mostly overlooked by industry because they sacrificed performance for memory safety in a time when RISC processors were demonstrating how much performance was available to simplified integrated circuit computers [11]. However, we can learn from these validated capability machines to build a comprehensive segment system for modern RISC processors.

3 A Capability Segment Model for RISC Processors

A segment mechanism that implements the capability model of safe, programmatic memory protection should have three properties:

- All memory accesses must be via segments.
- The program must be able to restrict its segments but not expand them.
- The program must be able to pass control between domains possessing different segments without granting additional segments to either domain.

The first property, the non-bypassable property, implies an additional layer of indirection before the page table that all memory requests must pass through.

The second property, the guarded manipulation property, requires that segment descriptors be distinguished from general purpose data by the processor hardware, not only in registers, but also in memory. This segment manipulation property also implies new instructions for manipulating segment descriptors, or at least special restrictions on general purpose instructions when manipulating segment descriptors.

The third property, the protected call gate property, suggests a special flow control instruction with the ability to safely leave a domain, as defined by a set of segments, and enter a new domain.

3.1 Our Instruction Set Architecture

We have chosen to implement our RISC capability model by extending the 64-bit MIPS instructions set. We selected MIPS because it has a well established 64-bit instruction set and adheres to a prototypical RISC philosophy.

We chose to implement the universal enforcement property by adding a segment table through which all virtual addresses are offset before reaching the page table. General purpose loads and stores are implicitly offset via a fixed segment in the table and instruction fetches are offset via another fixed segment. This fixed implicit offset model is very similar to the i286 segment model. We have also added a complete complement of new load and store instructions which allow explicit use of other segments in the table.

By storing segment descriptors in a distinct register file, separate from general purpose registers, we partially fulfill the guarded manipulation property by not allowing general purpose manipulation. We also added a dedicated set of segment descriptor manipulation instructions which only allow reducing the privilege of descriptors. We decided to protect segment descriptors in memory using tags on general purpose memory locations rather than using dedicated memory so that compilers could treat descriptors as they do pointers, including storing them on the stack and passing them as arguments.

We are able to fulfill the protected call gate property with a single-cycle instruction. We observe that a domain can be trusted to manually store its own state and to "unpack" its own state when it is called. Since any number of segment descriptors can be stored in a segment, it is only necessary to make a single segment available in a new domain for the entire domain to be "unsealed". To facilitate this, we allow segments to be sealed by a domain and passed to other domains to be used in protected calls. This simple primitive is sufficient to support protected procedure calls and is easily implemented as a standard single cycle instruction.

Possibly the most radical feature mentioned above is tagged memory, which implies an extra bit for each word in memory. We were able to implement this with a simple scheme which is practical in a traditional computer system without modifying the external memory hierarchy. The tags for DRAM locations are stored together in the top of DRAM and a small tag controller sits at the mouth of the memory controller on the processor and provides a tag for every memory request made by the core. Our segment descriptors are 256 bits wide and must be aligned in memory, thus we have an overhead of 1-bit for every 256-bits of data (i.e. 0.4%).

4 Proposed Evaluation of the Capability Segment Model

We have only begun to evaluate our segment implementation, but we hope to demonstrate that several key applications will benefit from our RISC capability segment model.

4.1 Protected Procedure Calls

We would like to thoroughly study the low-level performance of crossing between protection domains using our novel RISC style protected procedure calls. This will include various trust relationships: mutually untrusting domains and calls with asymmetric trust. Our methodology here will likely focus on performance and compare against x86 segmentation domain crossings as well as domain crossings using seperate address spaces.

4.2 Sandboxing

We would like to demonstrate that our capability segment model can conveniently and efficiently support application sandboxing. State of the art application sandboxing depends on process separation using TLB enforcement. Thus protected interactions which are logically procedure calls must be implemented as inter-process communication involving the operating system and two independent memory spaces. Allowing program components to run in a hardware enforced segment of the program's address space should allow a simpler communication model and more efficient use of the TLB.

We have experience with the Capsicum [20] sandboxing API in FreeBSD and have a version of gzip which uses Capsicum sandboxes. We plan to implement the same sandbox architecture using user space segments. We will note the complexity of the code modifications required for each approach and will then measure how performance scales with the number of sandboxes. We expect that a large number of sandboxes will place undue pressure on the TLB and cause performance collapse in the Capsicum case but that the segment model will scale more closely to the unprotected model. We have done an initial experiment which sandboxes libpng using both techniques. While we have a methodology for measuring performance and at least some methodology for usability, we are still working on a methodology for measuring relative security which will involve formal verification.

4.3 C/C++ Annotations

Segments in our model can be used as general purpose pointers with a limited range of modifications available. David Chisnall has extended Clang and LLVM to accept annotations that implement pointers as segments to ensure they are used according to programmer intent, including bounds checking and read, write, and execute property enforcement. This approach is designed to allow simple security enhancement of existing code bases in C which often have poor security properties. We plan to evaluate usability of the annotations by measuring the number of lines that needed modification for added security benefit compared to pervasive bounds checking, and by attempting to compare the clarity of resulting code.

5 Future Work/Collaboration Potential

Enforcing Program Structure in the Linker

Modern ELF files which are consumed by a linker to incorporate binaries into a running program specify segments of the program along with properties for each segment. It should be possible to build a custom linker which constructs a segment descriptor for each ELF segment and implements memory references as segment loads and stores.

Managed Language Runtimes

Flexible segment hardware should allow managed language runtimes to be simpler, faster, and more secure. A straight forward use of segment registers as pointers and of protected calls on invocations would provide strong hardware protection of managed language objects.

Static Analysis Assistance

Static analysis of computer programs for correctness has proven very promising. If segment manipulation was part of the user space code stream, static analysis may be able to make a breakthrough in reducing proof complexity.

6 Conclusion

We hope to demonstrate that a user space segment model is useful in the context of the modern software stack to enhance the security and correctness of computer programs. We also hope to demonstrate that a user space capability segment model is implementable in a modern RISC processor without breaking the principles of RISC design.

An FPGA implementation of the 64-bit MIPS processor will be available for download this year and we will welcome collaboration with researchers who would like to apply our capability segment model to their own challenges in engineering secure software and systems.

6.1 Acknowledgments

We would like to thank our colleagues - especially Peter Neumann, Jonathan Anderson, Ross Anderson, Nirav Dave, Ben Laurie, Steven J. Murdoch, Philip Paeps, Michael Roe, David Chisnall, Robert Norton and Khilan Gudka.

This PhD work is part of the CTSRD Project which is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Bibliography

[1] Chess, B.: Improving Computer Security using Extended Static Checking. In: Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on. pp. 160–173. IEEE (2002)

[2] Childs Jr, R., Crawford, J., House, D., Noyce, R.: A Processor Family for Personal Computers. Proceedings of the IEEE 72(3), 363–376 (1984)

[3] Cleveland, S.: x86-64 Technology White Paper. Tech. rep., Advanced Micro Devices (02 2002)

[4] Cowan, C., Wagle, F., Pu, C., Beattie, S., Walpole, J.: Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade. In: DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings. vol. 2, pp. 119–129. IEEE (2000)

[5] Denning, P.: Virtual Memory. ACM Computing Surveys (CSUR) 2(3), 153–189 (1970)

[6] Dennis, J.B., Van Horn, E.C.: Programming semantics for multiprogrammed computations. Commun. ACM 9(3), 143–155 (1966)

[7] Feiertag, R., Neumann, P.: The Foundations of a Provably Secure Operating System (PSOS). In: Proceedings of the National Computer Conference. vol. 48, pp. 329–334 (1979)

[8] Fotheringham, J.: Dynamic Storage Allocation in the Atlas Computer, Including an Automatic Use of a Backing Store. Communications of the ACM 4(10), 435–436 (1961)

[9] Frame, A., Turner, C.: Introducing New ARM Cortex-R Technology for Safe and Reliable Systems. Tech. rep., ARM (03 2011)

[10] Gudka, K., Watson, R., Hand, S., Laurie, B., Madhavapeddy, A.: Exploring Compartmentalisation Hypotheses with SOAAP. In: Workshop paper, Adaptive Host and Network Security (AHANS 2012) (2012)

[11] Hansen, P., Linton, M., Mayo, R., Murphy, M., Patterson, D.: A Performance Evaluation of the Intel iAPX 432. ACM SIGARCH Computer Architecture News 10(4), 17–26 (1982)

[12] Houdek, M., Soltis, F., Hoffman, R.: Ibm System/38 Support for Capability-based Addressing. In: Proceedings of the 8th Annual Symposium on Computer Architecture. pp. 341–348. IEEE Computer Society Press (1981)

[13] Levy, H.M.: Capability-Based Computer Systems. Butterworth-Heinemann, Newton, MA, USA (1984)

[14] Needham, R., Walker, R.: The Cambridge CAP Computer and its Protection System. Operating Systems Review pp. 1–10 (1977)

[15] Parrend, P., Frénot, S.: Classification of Component Vulnerabilities in Java Service Oriented Programming (SOP) Platforms. Component-Based Software Engineering pp. 80–96 (2008)

[16] Randell, B., Kuehner, C.: Dynamic Storage Allocation Systems. Communications of the ACM 11(5), 297–306 (1968)

[17] Saltzer, J., Schroeder, M.: The protection of information in computer systems. Proceedings of the IEEE 63(9), 1278–1308 (September 1975)

[18] Schroeder, M., Saltzer, J.: A hardware architecture for implementing protection rings. Communications of the ACM 15(3) (March 1972)

[19] Watson, R., Neumann, P., Woodruff, J., Anderson, J., Anderson, R., Dave, N., Laurie, B., Moore, S., Murdoch, S., Paeps, P., et al.: CHERI: A Research Platform Deconflating Hardware Virtualization and Protection. In: Workshop paper, Runtime Environments, Systems, Layering and Virtualized Environments (RESoLVE 2012) (2012)

[20] Watson, R.N.M., Anderson, J., Laurie, B., Kennaway, K.: Capsicum: Practical capabilities for Unix. In: Proceedings of the 19th USENIX Security Symposium. USENIX (August 2010)

Towards a Systematic Identification of Security Tests Based on Security Risk Analysis

Jan Stijohann[1] and Jorge Cuellar[1] (Supervisor)

Siemens AG, Germany,
firstname.lastname@siemens.com

Abstract. Today's security testing is not systematic much less standardized. In particular, there are no clearly defined criteria for selecting relevant tests. Thus different analysts come to different results and sound quality assurance is hardly possible. Literature suggests basing the choice and prioritization of tests on risk considerations but lacks a systematic approach for a traceable transition from abstract and business-oriented risk analysis into the concrete and technical security testing world. We aim at bridging this gap in two steps: The first one bridges between high-level and non-technical "business worst case scenarios" and less abstract "technical threat scenarios" using a technical description of the system and a systematic STRIDE-based elicitation approach. The second is a rule-based step that maps technical thread scenario to "test types", that is, to classes of tests that need to be adapted to the particular system under validation. Our method provides traceability for the choice for security tests and a standardized minimum quality assurance level.

Keywords: Security, risk driven testing, risk analysis, threat modeling

1 Introduction

Today's security testing is not a systematic or standardized process: a tester has neither clearly defined criteria for choosing or prioritizing possible tests, nor when to stop. The results of such security testing sessions depend on the tester's know-how, experience, intuition, and luck. Thus different people come to different results and a sound quality assurance is hardly possible.

Several standards and scientific papers suggest to base the choice and prioritization of tests on risk considerations. Yet only few explain how this can be done on a *technical* level, e.g. [8], and there is a lack of concrete guidelines on how to systematically select security tests based on *business*-oriented risk analysis [3].

However, if the risk analysis remains at a high level, it misses essential risk factors and is of little help for a subsequent security testing. If, on the other hand, it includes technical threats, such as "Buffer Overflow" or "SQL Injection", those choices seem arbitrarily taken, that is, independently of the unique, specific characteristics of the system under verification (SUV). It is not clear why they concern this particular SUV, nor why other similar threats with the same possible impact are ignored. For instance, in case of the buffer overflow, an

analyst may simply not consider "format string vulnerabilities", "double frees" or "null pointers", just because he is not aware of their existence, while "buffer overflow" is a common vulnerability that he knows.

Objectives This paper aims at bridging this gap between risk analysis and security testing, offering a systematic approach for a traceable transition from abstract, business-oriented risk analysis to the concrete and technical security testing world. The process must start from concrete technical information about the SUV and result in a set of tests that may still require some adaptation, but are manageable by experienced security testing experts. Moreover, it should be applicable in real-world industrial environments and provide a clear benefit for the security analyst, in terms of effort assurance, and transparency.

Outline of our Solution We propose a two-step-process for a systematic identification of security tests based on risk analysis:

1. The first one goes from high-level and non-technical "business worst case scenarios" to less abstract "technical threat scenarios" via a systematic *STRIDE*[1]-based elicitation approach. This step requires a sufficiently technical security overview, in the form of a Data Flow Diagram (DFD) of the SUV, annotated with security relevant information.
2. The second step derives security tests from the technical threat scenarios. It guides the analyst with rules that map patterns in the DFD to "test types", which are test templates which need to be instantiated, that is, adapted to the implementation, configuration and state of the SUV. In addition to those re-usable mapping rules, the selection of appropriate tests is supported by organizing the test types in a "test library" and tagging the entries according to the tested system element, the tested security property, the technology, and the sophistication of the test.

2 Current Work

2.1 Technical System Description

The technical system description captures and structures the security relevant technical aspects of the SUV in a comprehensive and systematic way. The resulting *security overview* is crucial for the transition from risk analysis results to security tests (see Section 2.2 and 2.3) and it provides the technical system information needed to identify and instantiate appropriate test types. The security overview should have the following properties:

Created by the Security Analysis Team. Design documents are often not suitable as a security overview as they tend to be overwhelming, out-dated, incorrect, incomplete, or they miss relevant security information. It is therefore judicious to let the risk analysis team create its own suitable, that includes

[1] The acronym is introduced as part of Microsoft's STRIDE threat modelling [7]; it stands for spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege.

sufficiently technical, security overview. Besides studying existing documents, the security analysts should consider interviews and white-board sessions with developers, as well as tool-supported approaches (as explained later on).

Sufficiently Technical. Examples of security relevant technical information that a security overview should consider in detail are: Data flow related aspects (including interfaces and trust boundaries), security controls (such as authentication, encryption, or input validation), sensitive information that must be protected, relevant design properties (protocols, sanitization/encodings, file permissions process privileges, etc.).

Generated with Tool Support. Manually creating a correct and sufficiently technical security overview can become time consuming and tedious. This is especially the case for complex and dynamically evolving software where no one has the complete overview. Reconnaissance tools, such as port scanners, network sniffers or static and dynamic analysis tools, can help to obtain and keep the overview. As a side effect, the tool-based generated results can be used to discover discrepancies with existing design documents and interview results.

Presented in a Syntactically Standardized Language. One possible standardized graphical language is given by Data Flow Diagrams (DFD) as used in [7], annotated with additional security-relevant information. DFDs are well suited for security analysis as they contain the interfaces that an attacker may use and describe how data, often the target of attack, moves through the system.

2.2 STRIDE-based Elicitation of Technical Threat Scenarios

A STRIDE-based elicitation of technical threat scenarios is the first step for the transition from high-level risk analysis into the practical security testing world. Starting point are short, informal, and non-technical descriptions of *business worst case scenarios* (BWCSs). Most risk analysis methods include the definition of BWCS or similar equivalents.

The security analyst examines which security violations of system elements could lead to the BWCSs. We call the tuple (*system element, violated security property*[2]) a *technical threat scenario* (TTS). The mappings of BWCSs to TTSs are created top-down (given a BWCS, examine which combination of TTSs could lead to it) and bottom-up (for each DFD element, check if a violation of any security property, in combination with other TTSs, could lead to a BWCS).

2.3 Mapping Technical Threat Scenarios to Test Types

The mapping of BWCSs to TTSs is only the first step towards security tests. The TTSs are still too abstract and need to be further concretized. For this purpose, we suggest the concept of *test types*. A test type is a template for a class of tests which a security analyst can instantiate into an executable test by adapting and completing it according to the implementation, configuration and state of the SUV.

[2] represented by one letter from the STRIDE acronym

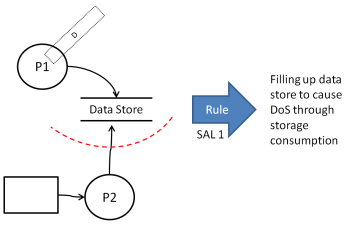
DFD-Pattern-based Rules One way to capture and leverage the security testing expertise required to derive appropriate test types from TTSs is via mapping rules. We suggest such rules to consist of the following elements:

- A pattern in an annotated DFD. Besides a mandatory TTS which includes the security property violation, the pattern can include additional system elements and further annotations.
- The level of sophistication for the security tests. It is determined by risk considerations such as the expected attacker and the desired assurance.
- A reference to the suggested test type that fits to the above characteristics.

Rules of this structure allow the test type derivation to become systematic and traceable. The mapping rules suggest an initial set of test types which helps to achieve a minimum quality standard.

Our DFD-patterns-based rules are inspired by EMC's "Threat library" [2], but in contrast our method is a) to be used during testing not development, b) uses explicit rules, c) considers SALs and security violations, and d) yields concrete test types instead of abstract threats.

Test Library The presented concept of mapping rules anticipated the idea of a re-usable collection of test types. We suggest that the entries of such a *test library* consist of a title, a textual description, and the information needed to be matched by mapping rules. The latter allows to filter the library entries according to the targeted system element, the security property to violate, the technology, and the sophistication of the test. This supports security analysts that want to go beyond the mere rule-generated minimum set of test types. Figure 1 shows an exemplary mapping rule and an excerpt of the current test library.



(a)

Name	SAL	Example Tools	Sys. Elem.	STRIDE
Reverse Engineering			**Process**	**I, E**
Pattern matching in hex editor	1	HxD	Process	I, E
Static binary analysis with disassembler	3	objdump	Process	I, E
Analysis of decompiled high-level code	2	.NET reflector	Process	I, E
Analysis of intermediate code	3		Process	I, E
Analysis with 3rd party tools	2	Sysinternals, strace, strings	Process	I, E
Dynamic binary analysis with debugger	2	OllyDbg	Process	I, E
Dynamic binary analysis with custom debugging scripts	3	IDA Pro scripts, pydbg	Process	I, E
Fuzzing	-		**Process**	**D**
Fuzzing with "point and shoot" tools	1	VolPer, 4n FTP Fuzzer	Process	D
Dumb fuzzing	1	/dev/random, netcat	Process	D

(b)

Fig. 1. (a) A rule maps a DFD pattern to a test type. (b) Excerpt of the test library.

2.4 Evaluation of Current Results

The steps of our method can be integrated in an ISO 31000 conform risk analysis process, provided that it estimates risk based on clearly defined threat scenarios. The suggested security overview is obtained in the **context establishment**

phase, together with the identification of non-technical assets and their security requirements. Based on this, the BWCSs are derived in the **risk analysis** phase and are then mapped to technical threat scenarios. The determination of expected threat agents including their estimated skill level is also part of the risk analysis phase. During the **risk estimation** phase, the likelihood of the technical threat scenarios is estimated and risk values for the BWCSs are assigned. The subsequent **risk evaluation** allows to prioritize the TTSs, which are then, as part of the **risk treatment**, covered by appropriate tests.

3 Future Work

3.1 Extending the Collection of Mapping Rules

We want to extend our current rule set in order to cover different SUVs from different application domains. For this purpose, we intent to proceed according to the following three-step procedure:

1. Identify classes of vulnerabilities to be covered. Appropriate sources are lists of threats and vulnerabilities such as CAPEC and CWE, and secure software development and security testing literature, such as [6], [4], and [5].
2. For each vulnerability class, analyse the technical context and identify suitable environment properties, in order to determine a reliable pattern that indicates the possible presence of the vulnerability.
3. Determine how the identified context information can be obtained, how it can be represented in form of a DFD pattern, and which (possibly new) test types match these patterns.

3.2 Tool Support

Our goal is to further automate the technical system description. We therefore plan to evaluate more advanced tools such as scriptable debuggers (e.g. pydbg, IDA Pro, Immunity Debugger), advanced dynamic analysis tools (e.g. WinApiOverride32, Microsoft Application Verifier), or operating system utilities (e.g. strace, eventlog). First steps are described in [8].

Once we have extended our rule base, a manual application of rules for the derivation of test types may become tiresome and inefficient. Our idea is to develop a tool that, similar as described in [1], processes annotated DFDs, detects the patterns and finally outputs adequate test types.

3.3 Further Evaluation

We plan to apply the presented method, together with a light-weight ISO 31000 conform risk analysis, in future real-world security assessments. The idea is to develop a questionnaire to capture observations after each assessment and thus support a systematic evaluation regarding benefits, drawbacks, practicability, areas for improvement, and inherent limitations.

We intend to analyse to what extend the tests, which have been identified using our method, satisfy the following requirements: 1) the security test addresses at least one BWCS. 2) it targets the proper system element and aims at violating the right security property, 3) it has the proper level of sophistication with respect to the expected threat agents 4) it reflects the technology, implementation and configuration of the SUV, and 5) the test priority is high enough with respect to the given time and budget.

4 Conclusions

This work proposes a tool-supported method to bridge the gap between a risk analysis and a corresponding security testing.

The presented method requires a security analysts to critically accompany the involved steps and adapt, possibly manually complement, and interpret the intermediate results. However, our method can guarantee a certain quality standard and, first and foremost, will make security testing more traceable for all involved parties including security testers, developers, and managers.

5 Acknowledgments

This work was partially supported by the EU Project SPaCIoS (FP7 257876, Secure Provision and Consumption in the Internet of Services) and NESSoS (FP7 256890, Network of Excellence on Engineering Secure Future Internet Software).

References

1. M. Abi-Antoun, D. Wang, and P. Torr. Checking threat modeling data flow diagrams for implementation conformance and security. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 393–396. ACM, 2007.
2. D. Dhillon. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security and Privacy*, 9(4):41–47, July 2011.
3. DIAMONDS Consortium; F. Seehusen (Editor). Initial methodologies for model-based security testing and risk-basedsecurity testing. http://www.itea2-diamonds.org/Publications/2012/index.html, Aug. 2012. Project Deliverable D3.WP4.T2_T3.
4. M. Dowd, J. McDonald, and J. Schuh. *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities*. Addison-Wesley Professional, 11 2006.
5. T. Gallagher, L. Landauer, and B. Jeffries. *Hunting Security Bugs*. Microsoft Press, 6 2006.
6. M. Howard, D. LeBlanc, and J. Viega. *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them*. McGraw-Hill Osborne Media, 9 2009.
7. M. Howard and S. Lipner. *Security Development Lifecycle*. Microsoft Press, 11 2009.
8. C. Wysopal, L. Nelson, D. D. Zovi, and E. Dustin. *The Art of Software Security Testing: Identifying Software Security Flaws*. Addison-Wesley Professional, 11 2006.

Penetration Test Tool for XML-based Web Services

Christian Mainka* Vladislav Mladenov† Juraj Somorovsky* Jörg Schwenk

Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
{christian.mainka, vladislav.mladenov, juraj.somorovsky, joerg.schwenk}@rub.de

Abstract

XML is a platform-independent data format applied in a vast number of applications. Starting with configuration files, up to office documents, web applications and web services, this technology adopted numerous – mostly complex – extension specifications. As a consequence, a completely new attack scenario has raised by abusing weaknesses of XML-specific features.

In the world of web applications, the security evaluation can be assured by the use of different penetration test tools. Nevertheless, compared to prominent attacks such as SQL-Injection or Cross-site scripting (XSS), there is currently no penetration test tool that is capable of analyzing the security of XML interfaces. In this paper we motivate for development of such a tool and describe the basic principles behind the first automated penetration test tool for XML-based web services named *WS-Attacker*.

Keywords: Penetration Test Tool, Web Service, XML Security, Signature Wrapping, Single Sign-On, WS-Attacker

1 Introduction

Service Oriented Architectures (SOAs) found the main idea for well-known and wide-spread technologies like Cloud computing and can be found in military services, e-Government, as well as in private and enterprise solutions. The main advantage of SOA is software reuse, modularization, and service out-sourcing. For its realization, well known interfaces have to be defined and the eXtensible Markup Language (XML) has become one of the key technologies for this task. Surrounded with the related W3C-standards such as SOAP, Web Services Description Language (WSDL) and XML Schema, XML is more than just a simple data description language – it is a full-featured platform-independent markup language.

The need for flexible security mechanisms in such architectures led to the development of addtional standards for securing SOA protocols. WS-Security relies on the already existing standards *XML Encryption* and *XML Signature* and applies them to SOAP-based web services. Moreover, WS-Trust is for establishing trust domains and WS-Policy/WS-SecurityPolicy are responsible for creating policies between communicating parties. Apart from web services, the Security Assertion Markup Language (SAML) OASIS Standard has gained increased popularity for Single Sign-On scenarios in enterprise web applications.

* The authors were supported by the Sec2 project of the German Federal Ministry of Education and Research (BMBF, FKZ: 01BY1030).

† The authors were supported by the SkIDentity project of the German Federal Ministry of Economics and Technology (BMWi, FKZ: 01MD11030).

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

Unfortunately, due to the complex design of these standards (e.g. XPath, XSLT, XML Signature, XML Encryption), their implementation has become very difficult. As a result of this, a lot of highly critical security flaws could be found in the processing of XML Signatures on SAML-based Single Sign-On frameworks [SMS+12]: eleven out of 14 systems were vulnerable to the XML Signature Wrapping (XSW) attack which was published by McIntosh and Austel seven years ago [MA05]. In the context of web services, a further work showed the effectiveness of this attack by breaking the Amazon EC2 as well as the Eucalyptus Cloud web interfaces [SHJ+11]. Even the confidentiality of XML Encryption protected messages could be annuled. Due to a bad usage of the CBC mode, the symmetric XML Encryption could be broken [JS11] and by applying Bleichenbacher' attack technique, the same authors also broke the asymmetric encryption [JSS12].

Besides the attacks on cryptographic primitives, there are also very efficient Denial-of-Service (DoS) attacks which abuse XML-specific characteristics. One example for this is the HashDoS attack which constructs special formed XML code in order to store XML attributes or namespace declarations in the same bucket of a vulnerable hash table and thus enormously slows down its processing[1]. Another example known as XML bomb uses XML entity declarations in a recursive way so that a message consisting of only a few KB will be expanded to several GB [JGHL07].

A huge problem from the security point of view is the complexity of the existing XML standards, which are often misunderstood. As a result of this, they are often not able to identify XML-specific security risks and therefore can not fix them. In the area of penetration testing tools for web applications customers can nowadays choose between several automated tools (or single components of such) for analyzing the security of systems in general or scanning for specific vulnerabilities, e.g. XSS and SQL-Injection. However, currently there is no known (commercial or open-source) software on the market that offers the ability to search and identify XML-specific weaknesses. This is our motivation to start working on a penetration test tool for web services.

2 Foundations

2.1 Attacking Web Services

The basic idea of a web service is to define an interface for message communication. The internal web service logic extracts the necessary information and forwards it to the underlying back-end. The problem of this approach is that the used XML standards for defining such an interface are very powerful and complex, thus a web service has mainly two different threat models:

Non-specific XML attacks abuse weaknesses in the back-end of an application, e.g Buffer Overflows or SQL-Injection.

Specific XML attacks exploit vulnerabilities in SOAP/web service and XML. They attack the XML parsing mechanism to enforce a DoS or build unexpected SOAP messages, e.g. change the SOAPAction header to confuse the web service logic.

It is important to mention that non-specific attacks are well known from web applications. However, compared to attacks such as XSS and SLQ-Injection, XML-specific attacks are totally new. They provoke the web service interface to behave unexpectedly by using XML-specific features. Currently, some penetration testing tools are able to handle web services, e.g. SOAP Sonar by Crosschecknetworks[2] or WSFuzzer by OWASP[3]. These tools support attacks such as SQL-Injection or XPath-Injection. Nevertheless, they do not handle all the XML-specific attacks. Therefore, we decided to develop our own penetration test tool called *WS-Attacker*[4] in order to fill the gap [MSS12].

2.2 XML Specific Attacks

A lot of XML-specific attacks exist and are known for a long time. Table 1 gives an overview on currently published attacks mainly taken from [JGH09]. Their classification, detailed information and even more attacks can be found on our website[5]. Due to the limited space, the next section will only focus on the XSW attack.

[1] CVE-2012-0841: `https://bugzilla.redhat.com/show_bug.cgiid=CVE-2012-0841`
[2] `http://www.crosschecknet.com/products/soapsonar.php`
[3] `https://www.owasp.org/index.php/Category:OWASP_WSFuzzer_Project`
[4] `http://sourceforge.net/projects/ws-attacker/`
[5] `http://ws-attacks.org`

XML Signature Wrapping	Attack on XML Encryption	Oversize Payload
Coercive Parsing	SOAPAction spoofing	XML Injection
WSDL Scanning	Metadata spoofing	Attack Obfuscation
Oversized Cryptography	BPEL State Deviation	Instantiation Flooding
Indirect Flooding	WS-Addressing spoofing	Middleware Hijacking

Table 1: Overview of existing XML-specific attack attacks.

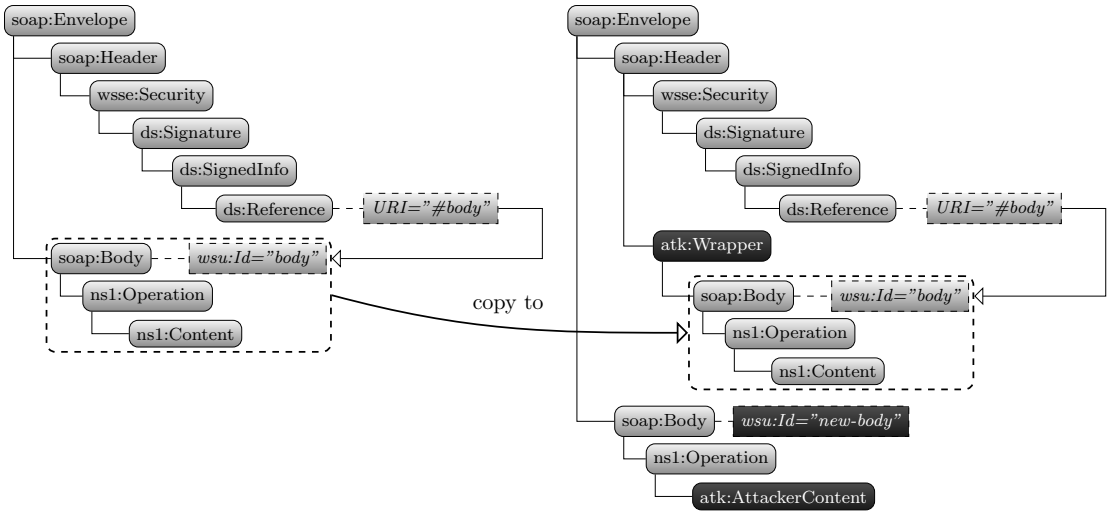


Figure 1: Basic XML Signature Wrapping scenario.

2.3 XML Signature Wrapping

XML Signature Wrapping (XSW) is an XML-specific attack first published by McIntosh and Austel in 2005 [MA05]. The very basic attack concept is shown in Figure 1.

Generally, the attack stems from the fact that the XML processing logic is mostly divided into two components: signature verification logic and application logic. The task of the signature verification logic is only to verify the signed content. In the depicted figure, the signature verification logic detects the signed content by only looking for any ID attribute with a specific value: `wsu:Id="body"`. After applying the attack as shown, it can still find the signed element in the attacker message, but it does not notice that it has moved. The application logic instead determines the element to process by just using the first element found as a child of the `<soap:Body>` element and ignores the ID attribute. Thus, the attacker's content is executed.

Note that different more complex attacks of this type exist [SMS+12, SHJ+11].

3 WS-Attacker's Task and XSW Attack's Complexity

The vast number of attacks on XML-based systems and the lack of an existing penetration test tool motivated us to develop *WS-Attacker*. The goal was to create a software solution which can be easily extended with any kind of XML-specific attacks. It is simple to use even for non-XML Security experts – which is realized by a easily understandable GUI which can be configured with only a few clicks – and can help to detect XML-specific vulnerabilities. Therefore, the user has to (1) load a WSDL, which identifies the web service endpoint, (2) send a test-request to the server to learn its *normal state* (behavior on untapered requets), (3) select the attack plugins, and (4) press a start button.

The need for such a penetration test tool is founded in the complexity of the attacks. Looking back to the XSW attack mentioned in the previous section as an example, Figure 2 visualizes its complexity. It is possible to have a large number of signed elements and each of it can be wrapped into a couple of positions within the XML document, e.g. located *somewhere* in the `<soap:Header/>`, or in the `<soap:Body/>`. Additionally, the wrapper can be placed as the first child, the last child, or somewhere in between. For each of this position, there can be additional adjustments (e.g. change the ID-value or keep it). The XSW attack can become even more complex

Figure 2: The complexity of the XSW attack.

when taking care of XPath based signatures [GJLS09] or the namespace injection technique [JLS09]. As a result of the different attack variants, a human attacker is not able to test all attack vectors.

This workflow clarifies that the attack performation by hand is nearly impossible. Besides incredible time consumption as a result of the different attack variants, a human attacker is not able to test all attack vectors.

Note that this is only an example for XSW attack, but this or a similar complexity can also be found on attacks on XML Encryption and XML DoS.

4 Future Work

In this section we give an overview of the known attacks on web service, which could be used to extend our framework.

4.1 XML-Specific Attacks

Our framework currently covers only a few of the attacks shown in Table 1. At the moment, there are already some existing attacks implemented, e.g. SOAPAction Spoofing and WS-Addresing Spoofing[6]. Even the powerful XSW attack can be automatically performed, including all attack variants and wrapping possibilities on ID-based signatures as well as on XPath-based systems. SAML over SOAP is also already implemented, and we are currently focused on browser-based SAML Single Sign-On as an extension of the WS-Attacker. However, the implementation of this extension is not trivial at all. Besides the XSW-attacks we want to integrate further tests regarding the configuration of the provider and already known bugs. Therefore, we need a very flexible and extensible software architecture able to generate dynamically SAML tokens. Furthermore, we require an evaluation logic analyzing the reaction of the tested system in response to the applied attack vectors. However, this evaluation is not a trivial issue due to the differences between the various systems accepting SAML tokens. Additionally, we are close before the release of XML-specific DoS attacks. The attacks on XML Encryption or the XXE (Xml eXternal Entity) attacks[7] are considered as our future work.

4.2 Beyond XML

Besides the XML-based services and protocols, other standards such as OpenID or OAuth became increasingly important in Single Sign-On scenarios. Moreover, current researches show the expanding usage of OpenID[8]. In addition to SAML, OpenID and OAuth are the most used protocols in the Cloud environment in order to authenticate users. For this reason their security became a part of common researches and has already been investigated by Wang et al. [WCW12]. They found critical security bugs in the authentication process, which allowed them to sign-in as an arbitrary user by misusing control flaws between Service Providers and Identity Providers like Facebook and Google. This work has been complemented by Sun and Beznosov [SB12]. However, none of these studies explicitly handles signature processing flaws at the Identity Providers. Thus, we see the automatic testing of OpenID and OAuth signature validation as a challenge in our future work, which could be included in our WS-Attacker framework.

In addition to the SOAP-based web service standards, many REST[9]-based web service interfaces support custom XML-based security mechanisms or follow the newest JSON security standards: JSON Web Signature [JRH12] and JSON Web Encryption [JBS12]. Jager et al. have already shown that their attacks on XML Encryption [JSS12] could be directly applied to the JSON Web Encryption standard. Automation and extension of these attacks could be considered as a next part of our future work.

[6] `http://ws-attacks.org`
[7] `http://www.agarri.fr/blog`
[8] `http://trends.builtwith.com/docinfo/OpenID`
[9] Representational state transfer

5 Conclusion

The threat of XML-based attacks has significantly increased. So does their application field: Besides web services, also Single Sign-On systems are attackable as latest researches have revealed [SMS+12]. This underlines the necessity of an automatic penetration test tool. Our solution – WS-Attacker – currently supports the first XML-specific attacks on web services, including the powerful XSW attacks with the majority of the known attack variants.

This paper gave an overview of the WS-Attacker framework and its basic functionalities. It sketched the future directions in the development of further XML-specific attacks, as well as of attacks beyond XML and web services. We believe that such an all-in-one solution will significantly help developers in finding vulnerabilities in their systems.

References

[GJLS09] Sebastian Gajek, Meiko Jensen, Lijun Liao, and Jörg Schwenk. Analysis of signature wrapping attacks and countermeasures. In *ICWS*, pages 575–582. IEEE, 2009.

[JBS12] M. Jones, J. Bradley, and N. Sakimura. JSON Web Signature (JWS) – draft-ietf-jose-json-web-signature-06, October 2012.

[JGH09] Meiko Jensen, Nils Gruschka, and Ralph Herkenhöner. A survey of attacks on web services. *Computer Science - R&D*, 24(4):185–197, 2009.

[JGHL07] Meiko Jensen, Nils Gruschka, Ralph Herkenhner, and Norbert Luttenberger. Soa and web services: New technologies, new standards - new attacks. In *Proceedings of the 5th IEEE European Conference on Web Services (ECOWS)*, 2007.

[JLS09] Meiko Jensen, Lijun Liao, and Jörg Schwenk. The curse of namespaces in the domain of xml signature. In Ernesto Damiani, Seth Proctor, and Anoop Singhal, editors, *SWS*, pages 29–36. ACM, 2009.

[JRH12] M. Jones, E. Rescorla, and J. Hildebrand. JSON Web Encryption (JWE) – draft-ietf-jose-json-web-encryption-06, October 2012.

[JS11] Tibor Jager and Juraj Somorovsky. How To Break XML Encryption. In *The 18th ACM Conference on Computer and Communications Security (CCS)*, October 2011.

[JSS12] Tibor Jager, Sebastian Schinzel, and Juraj Somorovsky. Bleichenbacher's attack strikes again: breaking PKCS#1 v1.5 in XML Encryption. In Sara Foresti and Moti Yung, editors, *ESORICS*, LNCS. Springer, 2012.

[MA05] Michael McIntosh and Paula Austel. XML signature element wrapping attacks and countermeasures. In *SWS '05: Proceedings of the 2005 Workshop on Secure Web Services*, pages 20–27, New York, NY, USA, 2005. ACM Press.

[MSS12] Christian Mainka, Juraj Somorovsky, and Jörg Schwenk. Penetration testing tool for web services security. In *SERVICES Workshop on Security and Privacy Engineering*, June 2012.

[SB12] San-Tsai Sun and Konstantin Beznosov. The devil is in the (implementation) details: an empirical analysis of oauth sso systems. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 378–390, New York, NY, USA, 2012. ACM.

[SHJ+11] Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, and Luigi Lo Iacono. All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces. In *The ACM Cloud Computing Security Workshop (CCSW)*, October 2011.

[SMS+12] Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, and Meiko Jensen. On breaking saml: Be whoever you want to be. In *21st USENIX Security Symposium*, Bellevue, WA, August 2012.

[WCW12] Rui Wang, Shuo Chen, and XiaoFeng Wang. Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services. In *IEEE Symposium on Security and Privacy (Oakland), IEEE Computer Society*, May 2012.

How do we effectively monitor for slow suspicious activities?

Harsha K. Kalutarage, Siraj A. Shaikh, Qin Zhou and Anne E. James
{kalutarh, aa8135, cex371, csx118}@coventry.ac.uk

Digital Security and Forensics (SaFe) Research Group
Department of Computing, Faculty of Engineering and Computing
Coventry University
Coventry, CV1 5FB, UK

Abstract As computer networks scale up in size and traffic volume, detecting slow suspicious activity, deliberately designed to stay beneath the threshold, becomes ever more difficult. Simply storing all packet captures for analysis is not feasible due to computational constraints. Detecting such activity depends on maintaining traffic history over extended periods of time, and using it to distinguish between suspicious and innocent nodes. The doctoral work presented here aims to adopt a Bayesian approach to address this problem, and to examine the effectiveness of such an approach under different network conditions: multiple attackers, traffic volume, subnet configuration and traffic sampling. We provide a theoretical account of our approach and very early experimental results.

1 Introduction

In the domain of computer security, an attacker may take days, weeks or months to complete the attack life cycle against a target host. This allows for such activity to blend into the network as noise. There is no clear definition to distinguish slow attacks from typical attacks. Of interest to us is any suspicious attempt to stay beneath intrusion detection thresholds. Detection of such 'low and slow' attacks pose a number of challenges. Simply storing and analysing all full content capture is not feasible due to computational constraints. Research addressing this area uses incremental anomaly detection approaches. Most of current incremental anomaly detection approaches have high rate of false alarm, are non-scalable, and are not fit for deployment in high-speed networks [MBK11]. Chivers et al. use Bayes formula to identify slow insider activities [CNS+09,CNS+10]; Phillip et al.'s work is similar [BBSP04]. In [BBSP04] user behaviour is profiled and used to identify those who require further investigations. Kalutarage et al. propose an aproach for cyber conflict attribution and reasoning in a Bayesian framework, together with concept of statistical normality [KSZJ12]. Streilein et al. use multiple neural network classifiers to detect stealthy probes [SCW02]. Evidence accumulation as a means of detecting slow activities is proposed in [T.H02]. Schultz et al. claim that profiling suspected insiders provides one of the best ways of reverse engineering an attacker [ER01].

2 How do we effectively monitor for slow suspicious activities?

Our aim is to study effective monitoring for slow suspicious activity that can be investigated further. We break our research objectives down to the following structure.

1. How do we effectively attribute slow suspicious activity to the source?
   The main goal here is to establish means of attributing such activity. At this stage we are interested in determining different possible methods to achieve this, and particularly investigating whether a Bayesian approach is feasible.

2. What effect does varying network parameters have over effective monitoring?

[2] In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

We are particularly interested in studying subnet size and traffic volume, and how that may effect our ability to distinguish such activity. We will draw from this network design principles for more effective monitoring.

3. How do we effectively detect the target of such activity?
   We acknowledge that the use of botnets and distributed sources makes it very difficult to attribute attacks. Of further interest is to determine the target of such activity. We will investigate methods to profile such nodes. Such methods need to be effective for scalable networks.

4. What effect does using sampling techniques has as a logging method?
   Traffic volumes will continue to increase. This makes it ever more difficult to process and effectively monitor slow activity. Since we are not detecting for strict traffic signatures, we wish to investigate traffic sampling methods and evaluate their suitability for security monitoring of slow attacks.

3 Research Methodology

We look at the problem as two sub problems: *profiling* and *analysis*. Profiling is the method for evidence fusion across space and accumulation across time, which updates the normal node profiles dynamically based on changes in evidence. Analysis is the method for distinguishing between anomalous and normal profiles using statistical normality. We propose to use elements of network flow data as input to our profiling method. Flow data contains network and port addresses, protocols, date and time, and amount of data exchanged during a session. We use a multivariate approach to analyse such records. So for example suspicious port scanning activity may have the following characteristics: a single source address, one or more destination addresses, and target port numbers increasing incrementally. When fingerprinting such traffic, we examine multiple elements and develop a hypothesis for the cause of behaviour on that basis. We use a Bayesion approach to achieve this.

3.1 Building the hypothesis

The posterior probability of the hypothesis H_k given that E, is given by the well known Bayes' formula:

$$p(H_k/E) = \frac{p\,(E/H_k)\,.p(H_k)}{p(E)} \tag{1}$$

Let H_k : hypothesis that kth node is an attacker, E_i is a flow record element and E =$\{E_1$=e$_1$, E_2=e$_2$, E_3=e$_3$,...,E_m=e$_m\}$ is the set of all suspicious evidence observed against node k during time t from m different independent observation spaces. Here $P(E)$ is the probability of producing suspicious events by node k, but on its own is difficult to calculate. This can be avoided by using the law of total probability. For independent observations, the joint posterior probability distribution can be obtained from (1) as:

$$p(H_k/E) = \frac{\prod_j p(e_j/H_i).p(H_k)}{\sum_i \prod_j p(e_j/H_i).p(H_i)} \tag{2}$$

To calculate the posterior probability of node k being an attacker $p(H_k/E)$, it is necessary to estimate:

1. the likelihood of the event E given the hypothesis H_i, $p(E/H_i)$ and,
2. the prior probability $p(H_i)$, where $n \geq i > 0$.

Assuming that prior and likelihoods are known, (2) facilitates to combine evidence from multiple sources (all E_is) to a single value (posterior probability) which describes our belief, during a short observation period, that node k is an attacker given E. Aggregating short period estimations over time helps to accumulate relatively weak evidence for long periods. This accumulated probability term, $\sum_t p(H_k/E)$ (t is time) known as profile value hereafter, can be used as a measurement of the level of suspicion for node k at any given time. These scores are converted into Z-scores for analysis.

A series of experiments have been conducted in a simulated environment to test the proposed approach. We use NS3 [NS311] to simulate our network and generate traffic patterns of interest, assuming a poison

arrival model. Each simulation is run for a reasonable period of time to ensure that enough traffic is generated (over one million events). If λ_s, λ_n are mean rates of generating suspicious events (where we only generate a subset of flow data elements including source and destination address and port numbers, and where suspicious activity is judged by unexpected port numbers) by suspicion and normal nodes respectively, we ensure maintaining $\lambda_s = (\lambda_n \pm 3\sqrt{\lambda_n})$ and $\lambda_n (\leq 0.1)$ sufficiently smaller for all our experiments to characterise slow suspicious activities which aim at staying beneath the threshold of detection and hiding behind the background noise.

3.2 Early Results

Early results of our work are promising: our approach is able to distinguish multiple suspicious nodes from a given set of network nodes as shown in Figure 1.

We model *detection potential* D as a function of subnet size S and traffic volume V, where $D = k.(\frac{V}{b^S})^{\frac{1}{2}}$, and where k is a constant, which demonstrates the effect of varying the subnet size over ability to detect effective monitoring. This effect is demonstrated in Figure 2. The effects of total traffic volume on detection potential are also demonstrated in Figure 3. Relevant details for these results could be found in [KSZJ12].

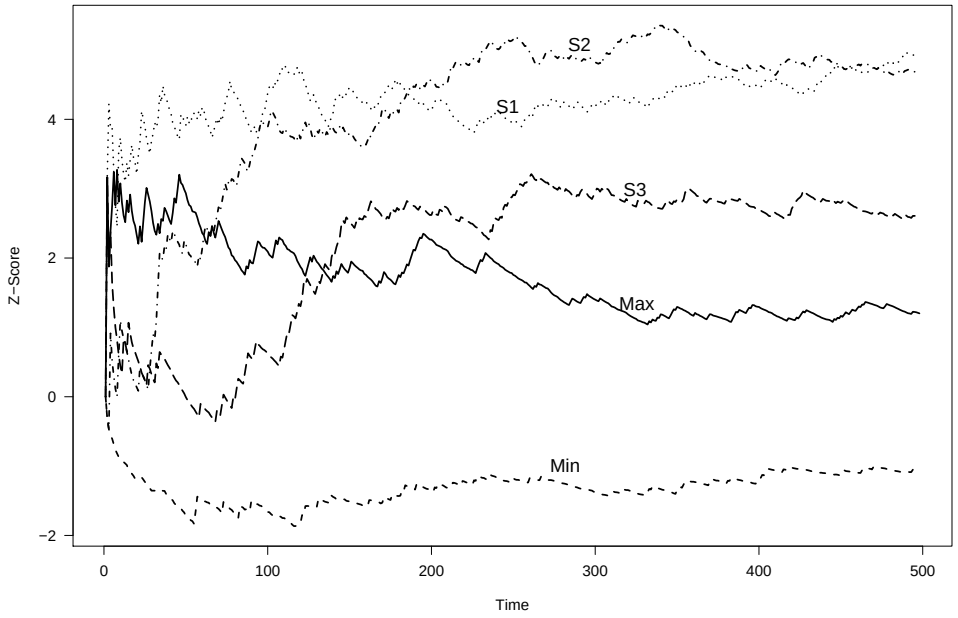


Figure 1: A Z-Score graph - for slow attack monitoring. S_i represents suspicious nodes. Min and Max represent the minimum and maximum Z-scores of normal nodes.

Our work aims to address the stated research goals by demonstrating how effective monitoring could be deployed in more realistic network topologies. We plan to continue with our experimental approach, and consolidate results towards the end to ensure a coherent and consistent picture emerges that is of practical value.

4 Contribution

This research aims to address a difficult problem. Monitoring infrastructures are overloaded both with data and tools. The question is: what do we with it? The difficulty is due to the increasing scale of networks, the diversity of user access provision to systems, the nature of suspicious activity and the corresponding need to monitor for serious attacks, and ultimately being able to effectively manage detection of intrusions.

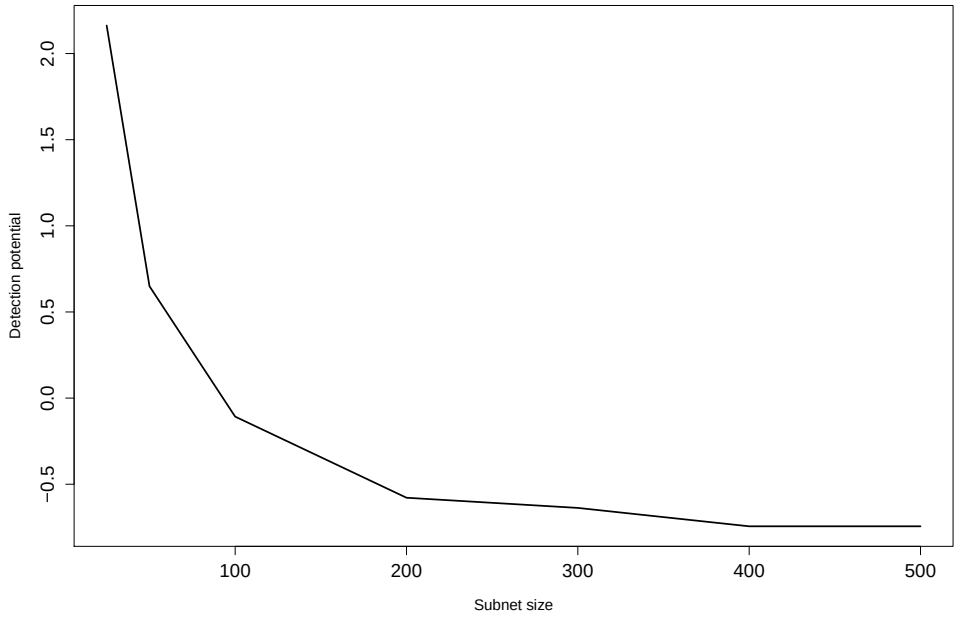


Figure 2: Smaller the subnet size, the better for detection⇒D $\propto \frac{1}{b^S}$, b is a constant.

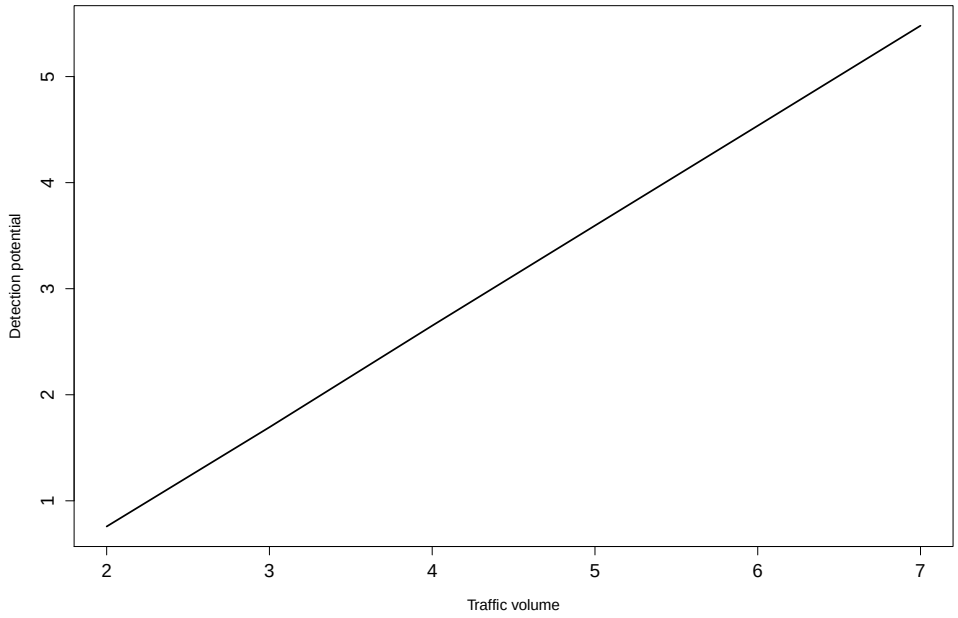


Figure 3: Higher the traffic volume, the better for detection⇒D $\propto V$.

Our ultimate goal is to offer a set of design principles and heuristics allowing for effective collection and analysis of data on networks. The first two research questions from Section 2 allow us to build defensible networks, where any source of suspicious activity could be detected effectively and quickly. This is about both better data analysis and network design. The third research question is inspired by related work investigating exposure maps [DOE06] and darkports [WvOK07], where we adapt our algorithm to profile target nodes for possible slow and suspicious activity. The underlying principle remains the same: we trade in state for computation. Ever increasing processing capacity increasingly makes this feasible. But traffic volumes indeed also pose a big challenge, and hence our final question is an attempt assess the feasibility of sampling traffic for analysis. This is also evidenced as feasible by some other work [BR12,PRTV10], and we propose to build on it.

Our aim is to remain domain agnostic. This allows for research to be applied at various levels, including better detection software, monitoring tools, and network design and configuration management solutions.

References

[BBSP04] Phillip G. Bradford, Marcus Brown, Bonnie Self, and Josh Perdue. Towards proactive computer-system forensics. In *In International conference on information technology: Coding and computing,IEEE Computer Society*, 2004.

[BR12] Karel Bartos and Martin Rehak. Towards efficient flow sampling technique for anomaly detection. In *Proceedings of the 4th international conference on Traffic Monitoring and Analysis*, TMA'12, pages 93–106, Berlin, Heidelberg, 2012. Springer-Verlag.

[CNS+09] Howard Chivers, Philip Nobles, Siraj Ahmed Shaikh, John Clark, and Hao Chen. Accumulating evidence of insider attacks. In *(MIST 2009) (In conjunction with IFIPTM 2009) CEUR Workshop Proceedings*, 2009.

[CNS+10] Howard Chivers, Philip Nobles, Siraj Ahmed Shaikh, John Clark, and Hao Chen. Knowing who to watch: Identifying attackers whose actions are hidden within false alarms and background noise. *Information Systems Frontiers, Springer*, 2010.

[DOE06] Whyte David, P.C.van Oorschot, and Kranakis Evangelos. Exposure maps: removing reliance on attribution during scan detection. In *Proceedings of the 1st USENIX Workshop on Hot Topics in Security*, pages 9–9, Berkeley, CA, USA, 2006. USENIX Association.

[ER01] E.E.Schultz and R.Shumway. Incident response: A strategic guide for system and network security breaches indianapolis. In *New Riders*, 2001.

[KSZJ12] Harsha K. Kalutarage, Siraj A. Shaikh, Qin Zhou, and Anne E. James. Sensing for suspicion at scale: A bayesian approach for cyber conflict attribution and reasoning. In *InProceedings of 4th International Conference on Cyber Conflict, NATO CCD COE*. NATO CCD COE Publications, Tallinn, June 2012.

[MBK11] M.H.Bhuyan, DK Bhattacharyya, and JK Kalita. Survey on Incremental Approaches for Network Anomaly Detection. *International Journal of Communication Networks and Information Security (IJCNIS)*, 3, 2011.

[NS311] NS3 Development Team. Ns3 discrete-event network simulator for internet systems, 2011.

[PRTV10] Antonio Pescap, Dario Rossi, Davide Tammaro, and Silvio Valenti. On the impact of sampling on traffic monitoring and analysis . In *Proceedings of 22nd International Teletraffic Congress (ITC) 2010*, pages 1–8, 2010.

[SCW02] William W. Streilein, Robert K. Cunningham, and Seth E. Webster. Improved detection of low-profile probe and novel denial-of-service attacks. In *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, 2002.

[T.H02] T.Heberlein. Tactical operations and strategic intelligence: Sensor purpose and placement. Technical Report TR-2002-04.02, Net Squared Inc, 2002.

[WvOK07] David Whyte, Paul C. van Oorschot, and Evangelos Kranakis. Tracking Darkports for Network Defense . In *Proceedings of Computer Security Applications Conference, 2007. ACSAC 2007.*, pages 161 – 171, 2007.

Global Design for Secure Socio-Technical Systems

Tong Li
Supervisor: John Mylopoulos, Fabio Massacci
University of Trento
via Sommarive 14, Povo(TN), Italy
tong.li@disi.unitn.it

Abstract.

Socio-Technical Systems (STS) consist of people, software, hardware and organizational units. The pervasiveness and complexity of STSs make security analysis both particularly challenging and especially critical. Traditional security analysis techniques that address security in a piecemeal fashion (e.g. only for software, or only for business processes) are insufficient for addressing global security concerns and have been found often to leave serious STS vulnerabilities untreated. In this proposal, we aim at developing a comprehensive framework that consists of concepts, techniques and tools for designing secure STSs. In our framework, a STS consists of organizational goals and security requirements, businesses and industrial processes through which requirements are satisfied, software applications that support those processes, and system infrastructure that supports both processes and applications. We intend to propose a systematic process to analyze and design each part of the STSs, and finally provide an all-round security design for STSs.

1 Introduction

The rapid developments of software systems, coping with information demands, have automated much of the manual tasks. There is no denying that the software systems are playing an increasingly significant role in large systems. Socio-Technical Systems (STS), as studied by Ropohl [18], introduce concepts which view systems from two perspectives—social and machine, stressing the reciprocal relationships between these perspectives. As STSs consist of many interweaving human and technical components, to design a STS, many complex sub-parts must be analyzed and well designed. For example, system organizational settings, business processes, software and infrastructures. A system can function correctly only if all the sub-components are well designed.

Security is a crucial and difficult issue for almost all systems. Ensuring security in STSs has been proved to be especially complicated [4], as addressing security means not only considering technical aspects of software, but also social aspects, such as the design of a business process. For example, if a business process, which issues customer orders, does not contain a step to check the authenticity of the order, the whole system will be vulnerable regardless of the security of other components. Moreover, the social aspect of a STS also requires to consider how do people interact with a STS. For example, if a security mechanism is too complex, stakeholders may close or bypass it, which leads the system to be vulnerable. In short, security protections for STSs are subject to the Bucket Effect: applying intensive security measures to a single component can not guarantee the security of the whole system.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

Unfortunately, most existing approaches address security requirements in a piecemeal fashion, merely considering security issues for a particular narrow part, i.e. software, physical infrastructure, or business process. We argue that the security designs of different parts in STSs may influence each other in certain ways. Currently, there are no formal approaches to connect all the design concepts in different system components and analyze their mutual impacts. Thus, no systematic, global means are proposed to design fully secure STSs.

To address this problem, we plan to develop a framework which consists of concepts, processes, and tools for designing secure socio-technical system by connecting different components with each other. This framework contributes in following aspects: 1) explicitly capture and represent causal relationships among different parts of a system to support the design in each part; 2) propose a systematic process to produce secure designs for each component — the synthesis of these designs is treated as the system design, which fulfills both organizational objectives and security requirements; 3) provide a systematic means to handle system changes, while continuously satisfy both organizational objectives and security requirements.

In the remaining part of this paper, we summarize the state of the art in Sec. 2, based on which we describe concrete research topics in Sect. 3. Sect. 4 shows our research approach that deals with the identified problems. Finally, the contribution of our work is summarized in Sect. 6.

2 Related Work

In this section, we will summarize related work on the topic of system security analysis and design. We are able to categorize many of these works according to the analytical perspectives they hold. Specifically, we consider following four perspectives: organizational settings, business processes, software applications and physical infrastructure. Work which falls outside of these classifications are discussed in a separate subsection.

There are a number of research proposals which focus on ensuring security of organizational settings through modeling interactions among social actors, as well as some social relationships (e.g. trust, delegation, authorization) [15, 7, 11].

Security analysis which focuses on business processes aims to represent the security requirements in business process models. Analysts can use these requirements to further design secure business processes [17, 9].

Most security efforts have focused on the technical aspects of software. There are many security analysis techniques which can represent either security attacks or security requirements, such as Abuse Case [12], Attack Tree [19]. In addition, there has been much investigation into systematic processes for security requirement engineering. For example, SQUARE [13] and SREP [14]. Moreover, security design methodologies are used to align software developments with security requirements, which capture a lot of security domain knowledge. Examples of security design methodologies include UMLsec [10] and Security Patterns [20].

Currently, there are few security approaches that focus on the infrastructure domain. Jürjens considers secure deployments in UMLsec [10] , which address security issues in the physical layer.

Mouratidis and Jürjens connect security designs between the organization perspective and the software perspective by combining Secure Tropos and UMLsec [16]. They provide a structured process to translate the results of organizational security requirements to pertinent software design elements to support secure software developments. However, they only transfer security requirements from the organization layer to the software layer, and do not provide feedback in the other direction.

Finally, there is another research branch that address human factors in STSs. Tarimo et al. [21] discuss the importance of security culture in the security analysis of STSs. They exploit the capability of organizational behavior theory, and explain how security cultures influence security requirements of STSs. In addition, Beautement et al. [3] argue that system stakeholders do concern with the costs and benefits introduced by security mechanisms, and a lot of security breaches arises due to overlooking the intentional factors of human. Thus, they propose Compliance Budget to understand human behaviors and further address the security balance problem.

3 Research Problem

Compared to technically-focused software systems, the socio-technical systems extend their system boundaries to include several related components, namely organization, business process and infrastructure. Although the proposed components may overlap with each other to some extent, we use them to ensure security designs cover all important aspects of STSs. Our work aims at analyzing all these essential components and investigating connections between them.

Organization component includes high-level system requirements. The concerns about organizational settings (e.g. the organizational hierarchies and policies) are considered in this domain.

Business process component is considered the core part of STSs, as all organizational requirements are met through business processes, and as business processes provide instructions for both humans and software on how to work together to achieve specific goals.

Software component provides concrete implementations for supporting the business process. In the meanwhile, it relies on the deployment of infrastructure.

Physical infrastructure component is the backbone of software applications. The performance of software applications highly depends on related infrastructures. In addition, they may also play a role in some activities in the business process.

Human factors are the major issues in STSs. Instead of addressing human influences in an independent dimension, we consider them together with each of the above components. We aim to consider human reactions together with related system designs, which facilitate understanding human intentions.

Based on the above components, our work will take the organizational goals and security requirements, which are in the organization component, as the inputs. While the outputs are a set of designs in the three left components, which work together to ensure security of the systems. In this sense, we define the designs in the software component as Software Requirement Specifications(SRS); the designs in business process component are Business Requirement Specifications(BRS) and the designs in infrastructure component are Infrastructure Requirement Specifications(IRS). Then the synthesis of all these designs is treated as the system specification, which satisfies both organizational goals and security requirements. Thus, the design of STS is defined as:

$$\text{System Design} = \text{BRS+SRS+IRS}$$

Current studies only take into account one of these parts and leave assumptions on other parts, which may not be held in the target system. We argue that these three parts of system designs are highly involved with each other. Without a global view on all the parts, the system under development will be vulnerable.

As a result, my research problem is summarized as developing a framework of concepts, processes, and tools for globally designing secure socio-technical systems by synthesizing designs in three different components, namely business process, software and physical infrastructure. Specifically, there are four research questions need to be addressed: 1) what are the interrelationships among the above components; 2) how to formally orchestrate analysis in different components to provide global security designs; 3) how to derive secure designs in each component; 4) how to adapt designs to handle system changes.

4 Research Approach

In this section, we propose a research approach, which addresses the security problems in each system component, in order to derive global secure designs for STSs.

4.1 Concepts of Secure STS

To specify system designs, we apply the following three concepts *Requirements (R), Specification (S) and Domain assumption (D)*, proposed by Zave and Jackson [22]. They have been formalized as follows: $D, S \vdash R$. For a specific domain this formula implies that the requirements can be satisfied by the specifications under relevant domain assumptions. A number of existing works make use of this conceptualization and formalism [8], but all of them take the machine domain by default. Our work aims to consider each component of STSs as a single domain. Each of them has its only requirements, domain assumptions, and specifications. In addition, we separate $SecurityRequirement(SR)$ from the concept $Requirement(R)$ in order to stress the security analysis in our research. As a result, the concepts can be represented by the formula $D, S \vdash R, SR$. This formula is further specified for each of the proposed components. For instance, $D_{bp}, S_{bp} \vdash R_{bp}, SR_{bp}$ represent the relevant concepts in the business process component. We structure the three components in a hierarchical way (Fig.1), which answers the research question 1.

Concretely, as shown in Fig.1, the organizational requirements are imported to the business process domain; the design of the business process domains specify the requirements for the software domain, as well as the infrastructure domain; the software domain should support business process, in the meanwhile shed light on the requirements for the infrastructure domain; finally, the infrastructure domain should support the design in other two domains. We use Tropos [5], a goal-based requirement modeling language, to represent requirements in all the layers. The specifications in all three layers are represented by BPMN [2], UML activity diagram and UML

deployment diagram [1] respectively. All of these models will be further extended with related security notations. In order to explicitly represent the semantics of each concepts, as well as their interrelationships, we aim to develop a formal ontology with Description Logic to capture all related knowledge (both system development and security engineering).

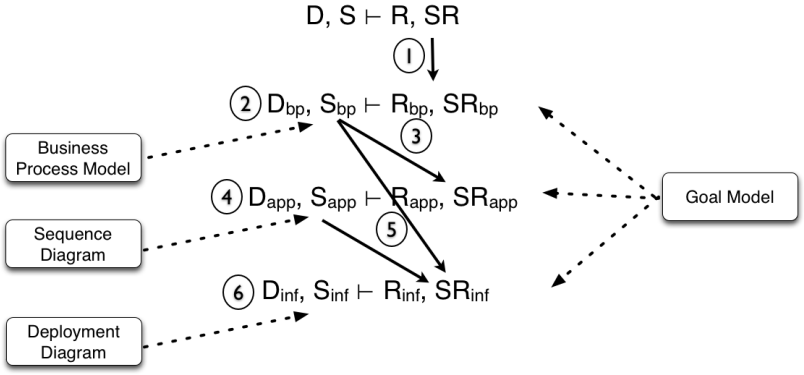


Fig. 1. Process for multi-layer system design

4.2 Secure STS Design Process

Based on the comprehensive ontology we are planning to build, we will further propose an intuitive process to orchestrate design processes in different layers(Fig.1), which address the research question 2. In each layer, its requirements are influenced by designs of other layers. In this process framework, analysis is conducted from the business process layer to the software layer, finally to infrastructure layer. It is worth noting that the proposed process is not like waterfall model, which is analyzed once for all, but an iterative process that incrementally addresses system designs. Through these analytical processes, the traceability links between layers could be derived, which are the key issues for global system designs.

The traceability links are built from specifications in one layer to requirements in other layers, not to specifications in other layers. In this sense, we do not directly impose designs from one layer to another, which indicates designs in each layer are only subject to requirements of that layer. The benefits from this separation include: 1) the requirement model plays as the mediator to connect designs in different do- mains, which clearly separate problems and solutions. By using the intermediary requirement model, the many-to-many mapping relationships between designs that are in different layers could be simplified as one-to-many relationships, which reduce the complexity of the design work and increase the adaptability of the system design. 2) By using the goal models to represent requirements, we can exploit the inherent advantages of the goal-based analytical techniques to facilitate the design work, such as the analysis of requirement satisfaction and alternative selection. As a result, this proposed framework could be used not only for developing new systems, but also for developing systems that are based on legacy systems. If existing designs of a legacy system cannot satisfy requirements imposed by previous layers, the analysis will backtrack to the previous layers and try to find another alternative.

There are six tasks identified in the Fig. 1, which outline the process for deriving an all-round system design. Concretely, the tasks 1,3 and 5 focus on how to derive general requirements and security requirements from upper layers. To this end, mapping mechanisms among layers will be defined to assist the derivation of requirements. The tasks 2,4 and 6 focus on how to derive secure designs that fulfill both general requirements and security requirements. We intend to carry out risk-based security analysis processes to derive security designs by adopting security patterns. This general analytical process will be customized for each layer according to their domain specific features. By introducing such a design process we satisfy the research question 3.

4.3 Evolution of Secure Design

Changes are inevitable for most systems, and these changes must be handled in a way that ensures the continued satisfaction of system requirements. The essence of our work is to combine concepts in multiple layers, based on which we are able to identify designs in different layers which are effected by change requests.

In front of changed requirements, we will incrementally evolve the system rather than re-design from scratch. Thus, appropriate strategies should be applied to promote the incremental design process. Referring to the

work[6]. We consider three kinds of approaches, namely minimal efforts, minimal changes and maximal familiarities.

4.4 Approach Illustration

The expected contribution of this research is providing a systematic means to globally design a secure STS. In this part, we use a simple example to show the ideal system development process. A smart grid real-time pricing scenario is used as an example for illustration. Due to the space limitation, we only focus on a particular part of this example, which is highlighted by dotted rectangles.

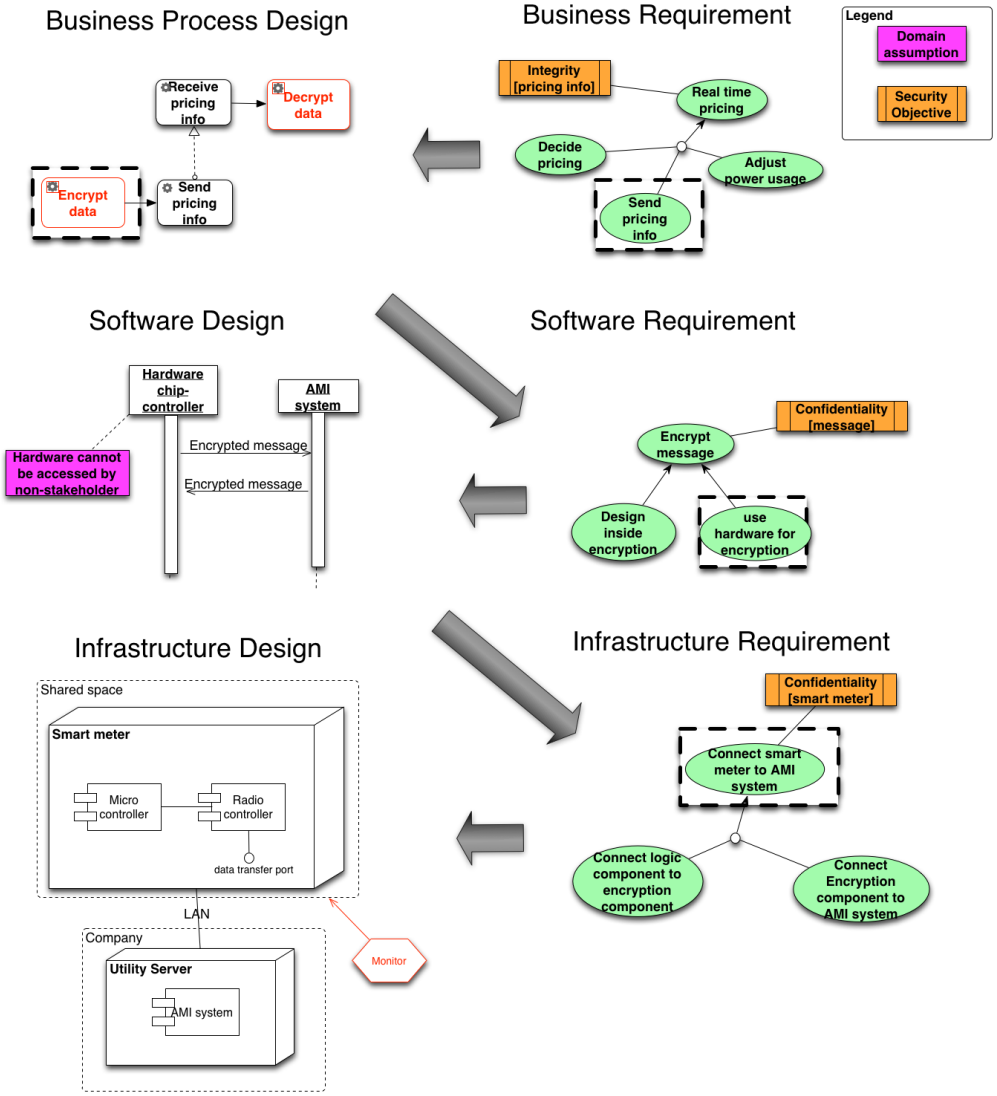


Fig. 2. Illustration of global design for secure STS

The whole design process is shown in Fig.2, which stick to the process described in Fig.1. Except for the notations introduced by existing work(e.g. Tropos, BPMN, UML), we add another two concepts, domain assumption and security objective, into the design process. Domain assumption represents the assumption that are taken during system design, and security objective represents security properties that have to be held during a goal achievement process. Fig.2 shows an intuitive order for system design, which could be conducted backwards and iteratively as necessary.

When there is a change happens at arbitrary part of the system design, the system will evolve accordingly to continuously satisfy the organizational goals. Fig.3 represents a redesign process via traceability links. Concretely, the change first cause a backwards propagation as shown in the upper part of Fig.3. The propagation will be

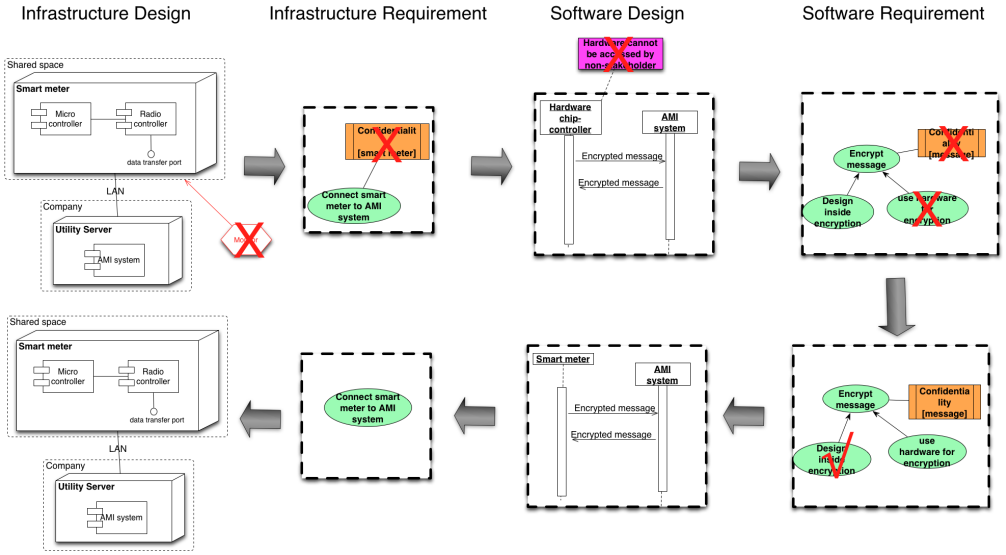


Fig. 3. Illustration of a system evolution according to a change

stopped until the change could be handled, e.g. find an alternative way to replace the unsatisfied requirement. After that, the system will forward the new design to related parts to "implement" it.

5 Evaluation Plan

Our framework aims to support the secure system design for socio-technical systems, and it is supposed to provide an all-around solution within comparatively efficient processes. To ensure the effectiveness and efficiency of the proposed framework, we should follow an evaluation methodology to examine the practical value of our work.

Tool support We are intended to develop a CASE tool, which can support all the steps of our analytical framework. Enhanced by this tool, the system design process could be done in a semi-automatic way. For the task 3 and 5, the tool can analyze the designs in upper layers and generate potential requirements, which will be further confirmed and modified by human. For task 2, 4 and 6, the tool will assist the security analysis processes to derive security design in each layer. In addition, the tool is able to analyze the ripple-effects of specific changes. All the design fragments influenced by the changes will be presented by the tool, and the customer will take the final decisions about which part should be changed.

Case study We will apply our design methods to a real socio-technical system, such as smart grid and electrical healthy system. The key point of the case study is to take out the theoretical method from laboratory to practice in order to check whether it works in reality. To this end, a number of factors need to be considered during the case study, such as the background of the method adopter, the time span for learning the method, the help or guide provided by the method designer. We need to exploit the influential factors as many as possible to provide an in-depth evaluation for our method. In addition, if it is possible we are planning to compare our method with other related work on the same case study to examine the advantages and disadvantages of our method.

6 Contribution

In summary, satisfaction of the proposed research goals via execution of the described plans will contribute to the state of the art by providing:

1. A conceptual multi-layer framework to understand the secure socio-technical system in a comprehensive way, specifically to understand the interactions among different system components.
2. A methodology to globally design secure socio-technical systems, which satisfy both organizational objectives and security requirements, either based on an existing system or for a new system. The methodology is aimed to enable the evolution of system design to manage requested changes, while continuously fulfilling organizational objectives and security requirements.
3. A tool that supports all the analytical steps of the proposed methodology. The tool will allow us to carry out an evaluation to examine the practical value of our work.

References

1. Unified Modeling Language. http://www.omg.org/spec/UML/2.1.2/, 2003.
2. Business Process Modeling Notation. http://www.omg.org/spec/BPMN/2.0/, 2011.
3. A. Beautement, M. A. Sasse, and M. Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 workshop on New security paradigms*, NSPW '08, pages 47–58, New York, NY, USA, 2008. ACM.
4. R. P. Bostrom and J. S. Heinen. STS Perspective MIS Problems and Failures : A Socio-Technical Perspective PART I : THE CAUSES. *MIS Quarterly*, 1(3):17–32, 1977.
5. P. Bresciani, A. Perini, P. Giorgini, and F. Giunchiglia. Tropos: An agent-oriented software development methodology. *Agents and Multi-Agent*, 2004.
6. N. A. Ernst, A. Borgida, and I. Jureta. Finding incremental solutions for evolving requirements. *Proceedings of 19th IEEE International Requirements Engineering Conference (RE)*, 0:15–24, 2011.
7. P. Giorgini, F. Massacci, and N. Zannone. Security and trust requirements engineering. In *Foundations of Security Analysis and Design III*, volume 3655 of *Lecture Notes in Computer Science*, pages 237–272. 2005.
8. C. Haley, R. Laney, J. Moffett, and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. *Software Engineering, IEEE Transactions on*, 34(1):133–153, 2008.
9. P. Herrmann and G. Herrmann. Security requirement analysis of business processes. *Electronic Commerce Research*, 6(3-4):305–335, Oct. 2006.
10. J. Jürjens. UMLsec: Extending UML for Secure Systems Development. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, volume 2460 of *Lecture Notes in Computer Science*, pages 412–425. 2002.
11. L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proceedings IEEE international conference on requirements engineering (RE'03)*, volume 3, pages 151–161, Monterey, California, 2003.
12. J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *15th Annual Computer Security Applications Conference, (ACSAC'99) Proceedings.*, pages 55–64. IEEE, 1999.
13. N. R. Mead and T. Stehney. Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes*, 30(4):1, July 2005.
14. D. Mellado, E. Fernández-Medina, and M. Piattini. A common criteria based security requirements engineering process for the development of secure information systems. *Computer Standards & Interfaces*, 29(2):244–253, Feb. 2007.
15. H. Mouratidis and P. Giorgini. A natural extension of tropos methodology for modelling security. In *the Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002)*, Seattle-USA,, 2002. Citeseer.
16. H. Mouratidis and J. Jurjens. From Goal-Driven Security Requirements Engineering to Secure Design. *International Journal of Intelligent System*, 25(8):813–840, 2010.
17. A. Rodriguez, E. Fernandez-Medina, and M. Piattini. A BPMN Extension for the Modeling of Security Requirements in Business Process. *IEICE transactions*, E90-D(4):745–752, 2007.
18. G. Ropohl. Philosophy of socio-technical systems. *Society for Philosophy and Technology*, 4(3):59–71, 1999.
19. B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.
20. M. Schumacher. Security patterns and security standards. In *Proceedings of the 7th European Conference on Pattern Languages of Programs (EuroPLoP), July*, 2002.
21. C. N. Tarimo, J. K. Bakari, L. Yngström, and S. Kowalski. A social-technical view of ict security issues, trends, and challenges: Towards a culture of ict security - the case of tanzania. In *ISSA*, pages 1–12, 2006.
22. P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, Jan. 1997.

Towards a Systematic Literature Review on Secure Software Design

Alexander van den Berghe Riccardo Scandariato Wouter Joosen

{firstname.lastname}@cs.kuleuven.be
iMinds-Distrinet, Department of Computer Science, KU Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

In recent years numerous researchers have proposed a wide variety of approaches to incorporate security concerns into software design. Unfortunately a systematic literature review (SLR) providing a detailed overview of the state of the art and defining interesting research opportunities is lacking. This creates an extra barrier for (new) researchers to enter the domain and contribute to it. We describe a procedure for an SLR aimed at minimizing this barrier. By providing this procedure we first hope to receive feedback on it and trigger a discussion. Second, the availability of this procedure is useful when updating the SLR with approaches that will emerge after its initial performance.

1 Introduction

Students starting a PhD must typically overcome two challenges before they can start contributing to a particular domain. On one hand they must define their own "niche" within the domain. On the other hand they must establish a good understanding of the state of the art. These challenges are most often tackled by studying the available literature. A systematic literature review (SLR), introduced into software engineering by Kitchenham and Charters [KC07], is a structured manner for executing this task.

We are performing an SLR concerning the incorporation of security concerns in software design. This domain has grown rapidly in recent years, with numerous researchers proposing a wide variety of approaches. Unfortunately these approaches are developed mostly independent from each other and focus on different security properties. This results in a complex tangle of different approaches, creating an extra barrier for (new) researchers to enter the domain and contribute to it.

The first objective of this SLR is to untangle the domain by providing a detailed overview of the current state of the art, useful to both new and experienced researchers. Second we aim at discovering the gaps in current research and thus define interesting research opportunities. In this paper we describe in detail the procedure defined for the SLR. However, the actual results of the SLR are not discussed here. By providing this procedure we first hope to receive feedback on it and trigger a discussion. Second, the availability of this procedure allows everyone to continuously update the SLR with approaches that will emerge after its initial performance.

The remainder of this paper is organized as follows. Section 2 discusses in detail the procedure of the SLR. Section 3 shortly describes related work. Section 4 concludes the paper.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

2 Systematic Literature Review

This section introduces the procedure we defined for the SLR. First, we describe the research questions and discuss how they relate to our goals. Second, we describe the criteria for scoping the relevant research works. Third, we describe the strategy to retrieve the relevant research works. Fourth, we describe how the research works are analyzed to provide answers to the posed research questions.

2.1 Research Questions

We define four main research questions (RQ's), shown below, for the SLR.

RQ1: What security properties are supported during software design?

RQ2: Is a representation of the security properties supported?

RQ3: Is an analysis of the security properties supported?

RQ3.1: Is the supported analysis precise?

RQ3.2: Is the supported analysis white hat or black hat?

RQ4: What evaluation is provided for the proposed approach?

To avoid ambiguity some terms in these research questions require a more precise description. *Software design* refers to both architectural and detailed design. *Security properties* includes properties such as integrity and logging. The full set of properties defined for the SLR is discussed later. *Supporting* a security property means providing the possibility to represent and/or analyze it. *Representing* a security property covers every explicit representation, graphical or textual, of a security property. *Analyzing* a security property means verifying whether it is correctly enforced according to the system's security policy, which is the collection of all security requirements. An analysis is considered *precise* if a developer can algorithmically perform it by following the steps in its description. Note that whether an analysis method is precise or not is irrelevant of any tool support.

The first three research questions allow to construct an overview of the state of the art. Furthermore, they allow to discover research opportunities concerning security properties that are not or barely addressed. The fourth research question uncovers approaches lacking evaluation, which are opportunities for empirical research.

2.2 Inclusion and Exclusion Criteria

The inclusion and exclusion criteria of an SLR delimit which research works are considered relevant. Papers are **included** if they adhere to at least one of the following criteria:

- The paper represents security properties in software design or
- analyzes security properties in software design or
- models attacks or threats in software design or
- evaluates a paper included by a previous criterion.

Papers are **excluded** if they adhere to at least one of the following criteria:

- The paper only mentions security as a general introductory term or
- is not available as a full version, only an extended abstract or presentation, or
- is a duplicate of an included paper or
- is superseded by an included paper or
- is published more then ten years ago.

If duplicate papers are encountered only the most recent or most extensive version is included. We defined a scope of ten years because approaches described in older papers either have been developed further, and are thus included through later papers, or have most likely become outdated for current technology.

The inclusion or exclusion of a paper is decided in two phases. First, during the *initial selection phase* the abstract, title and keywords of a paper are evaluated against above criteria. In case of doubt the conclusion of the paper is also consulted. Second, during the *final selection phase* the included papers are fully read and their inclusion is re-evaluated. Each paper is evaluated by one participant of the SLR while another participant validates this evaluation. If participants disagree on the inclusion or exclusion of a paper consensus should be reached through a discussion between all participants.

2.3 Search Strategy

A search strategy defines how to retrieve relevant research works. In order to achieve maximal coverage our search strategy consists of three complementary methods: a digital library search, a manual search and snowballing.

First, we searched **digital libraries** by means of a search string containing relevant terms. Table 1 contains an overview of the selected digital libraries. In our opinion this selection covers a sufficiently large amount of the defined domain and including more libraries would result in too much overhead. Furthermore the selected libraries provide extensive search functionality, allowing complex search strings. We considered using Google Scholar but due to technical limitations it was not included.

Table 1: Digital Libraries targeted with a search string

Library	Website
ACM	`https://dl.acm.org/`
CiteSeerX	`http://citeseerx.ist.psu.edu`
IEEExplore	`http://ieeexplore.ieee.org`
Springerlink	`http://link.springer.com`
ISI Web of Science	`http://apps.webofknowledge.com`
Compendex	`http://www.engineeringvillage2.org/`

An initial search string was constructed by selecting terms from a manually composed set of highly relevant papers. This search string was fine-tuned to reduce the number of irrelevant papers using the top 100 results of ACM and CiteSeerX. To avoid an explosion in the number of results the search string is only queried over the abstract, keywords and title. Due to space constraints we do not describe the final search string here[1]

Second, we performed a **manual search** of papers published in relevant conferences and journals. Tables 2 and 3 contain an overview of respectively the selected conferences and journals. In our opinion this is a representative selection of venues for the defined research domain.

Table 2: Conferences selected to be manually searched

Name	Acronym
European Conference on Object-Oriented Programming	ECCOOP
International Symposium on Engineering Secure Software and Systems	ESSoS
International Conference on Software Engineering	ICSE
International Symposium on Architecting Critical Systems	ISARCS
International Conference on Model Driven Engineering Languages and Systems	MODELS
Working IEEE/IFIP Conference on Software Architecture	WICSA
European Conference on Software Architecture	ECSA

Third, we performed the so-called **snowballing** method, both forward and backward, on included papers.

[1] More information can be found at `https://people.cs.kuleuven.be/alexander.vandenberghe/search-string.html`.

Table 3: Journals selected to be manually searched

Name	Acronym
Journal of Systems and Software	JSS
Software and Systems Modeling	SoSyM
Transactions of Software Engineering	TSE

2.4 Quality Assessment, Data Extraction and Synthesis

The research questions are answered by analyzing the included approaches[2]. First, the **quality assessment** uses a quality questionnaire, shown below, to score each approach. Allowing one to rank the approaches relative to each other.

1. How many papers are published for the approach?
2. How is the approach evaluated?
   - Industrial case study
   - Researcher or student case study
   - Toy example
3. How much tool support is available for the approach?
   - Full tool support
   - Partial tool support
   - No tool support

For the first question an approach is awarded one point per paper included in the SLR. For the second question an industrial case study awards two points, a researcher or student case study awards one point and a toy example awards half a point. Points are awarded per unique instance of an evaluation. If different papers describe the same evaluation points are awarded only once. If for example multiple papers for one approach describe the same toy example only half a point is awarded. For the third question full tool support awards one point whereas partial tool support awards half a point. Full support means one or more tools support the creation of all notational elements provided by the approach and its analysis method can be performed without user intervention. If only part of the approach is supported by one or more tools (e.g., a modeling tool is available but analysis must be performed manually) it is considered to have partial tool support.

Second, **data extraction** is performed using a taxonomy we defined for the SLR. Each approach is classified over three dimensions: security, software engineering and evaluation.

The *security dimension*, shown in Figure 1, provides essential data concerning the first three research questions. Besides this data any other security artifacts (e.g., test data) constructed by an approach can also be listed.

The *software engineering dimension*, shown in Figure 2, allows to situate approaches within the development process as a whole. Furthermore it allows comparing approaches based on their applicability in different development phases or domains and thus further elaborates the intended overview.

The *evaluation dimension*, shown in Figure 3, provides data for the fourth research question. This dimension allows to compare both the evaluations for one approach as well as the evaluation for different approaches.

Third, **data synthesis** summarizes the data obtained during data extraction to provide answers to the posed research questions. The first three research questions can be answered by tabulating the data extracted for the security and software engineering dimension. The fourth research question can be answered by tabulating the data extracted for the evaluation dimension.

[2]Since we want to analyze each approach as a whole, we group all papers concerning one approach together instead of analyzing each paper individually.

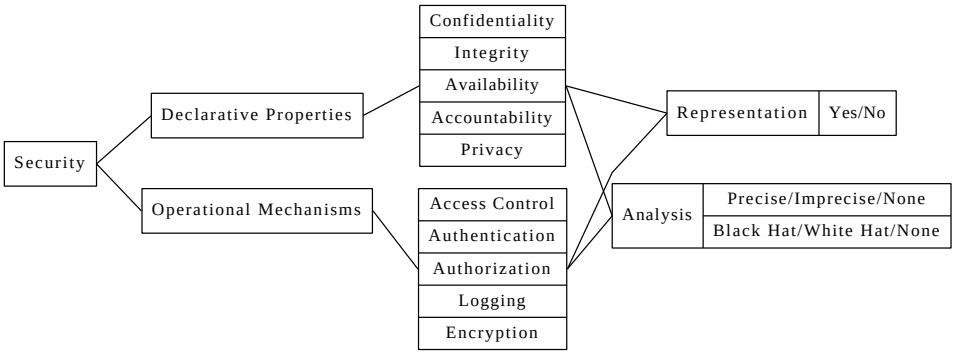


Figure 1: The security dimension classifies each approach based on which security properties it supports and how these properties are supported.

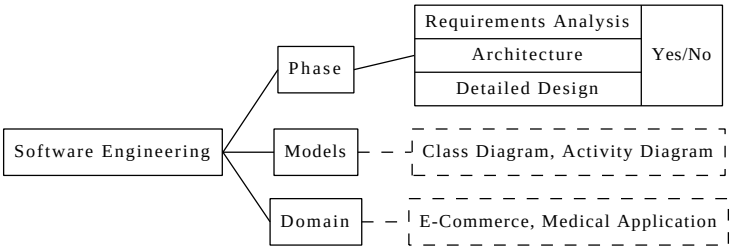


Figure 2: The software engineering dimension classifies each approach based on supported development phases, models and domains. Dashed rectangles illustrate example values.

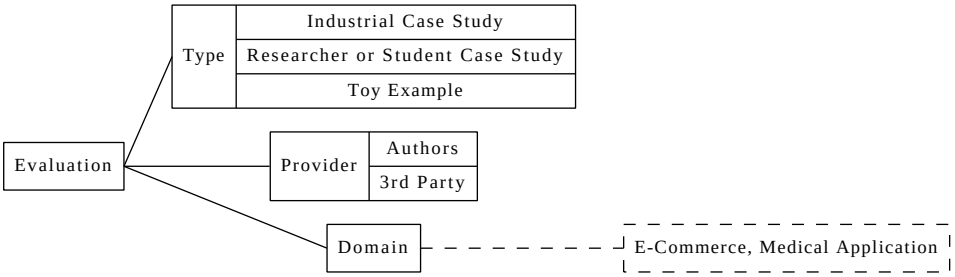


Figure 3: The evaluation dimension classifies each evaluation for an approach based on its type, who provides it and in which domain it is situated.

3 Related Work

In recent years studies comparable to the SLR proposed in this paper have been performed. These studies can be divided into three categories: SLR's, comparisons and surveys.

First, the **SLR** category contains studies following the guidelines by Kitchenham and Charters. Jensen and Jaatun [JJ11] study security in model-driven development. Due to their focus on code generation the scope of the SLR is rather narrow. Furthermore, no explicit comparison of included approaches is provided.

Second, the **comparison** category contains studies only comparing selected approaches. The conclusions from such studies are useful but provide little information for the domain as a whole. Matulevičius and Dumas [MD10] compare two approaches for their applicability to role-based access control.

Third, the **survey** category contains studies providing an overview of the domain, with optionally a comparison. These studies are not performed in a systematic manner making them difficult to update.

Dehlinger and Subramanian [DS06] survey aspect-oriented approaches for designing and implementing secure software. The included approaches are only individually evaluated without comparison between them. Jayaram and Mathur [JM05] cover a broader scope by surveying all types of approaches but focus mainly on the requirements phase. The authors provide no comparison and only general possible research directions.

Villarroel et al. [VFMP05] not only provide an overview but also compare the surveyed approaches. The authors use Khwaja and Urban's [KU02] comparison framework, which lacks security-specific criteria. The resulting comparison thus does not provide an adequate overview of security in software design. Kasal et al. [KHN11] solve this problem by defining their own evaluation taxonomy, inspired by Khwaja and Urban's framework. The authors define, among others, formality and security mechanisms as evaluation dimensions.

Dai and Cooper [DC07] evaluate and compare approaches based on supported security properties, used modeling notations, analysis support and examples. But they do not explicitly define their evaluation taxonomy.

4 Conclusion

In this paper we described in detail a procedure for a systematic literature review (SLR) concerning the incorporation of secure concerns in software design. With such an SLR we aim to provide a detailed overview of the state of the art and define interesting research opportunities. By making the procedure available we first hope to receive feedback and trigger a discussion. Second, the availability of this procedure allows everyone to continuously update the SLR with approaches that will emerge after its initial performance. Hopefully resulting in the continuous availability of an up to date SLR aiding researchers in entering and contributing to the domain.

Acknowledgment

This research is partially funded by the Research Fund KU Leuven, and by the EU FP7 project NESSoS. With the financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE).

References

[DC07] L. Dai and K. Cooper. A Survey of Modeling and Analysis Approaches for Architecting Secure Software Systems. *International Journal of Network Security*, 5(2):187–198, 2007.

[DS06] J. Dehlinger and N. Subramanian. Architecting Secure Software Systems Using an Aspect-Oriented Approach: A Survey of Current Research. Technical report, Iowa State University, 2006.

[JJ11] J. Jensen and M.G. Jaatun. Security in Model Driven Development: A Survey. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 704 –709, aug. 2011.

[JM05] K. R. Jayaram and Aditya P. Mathur. Software engineering for secure software - state of the art: a survey. Technical Report CERIAS 2005-67, Purdue University, 2005.

[KC07] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[KHN11] K. Kasal, J. Heurix, and T. Neubauer. Model-driven development meets security: An evaluation of current approaches. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–9, jan. 2011.

[KU02] Amir A. Khwaja and Joseph E. Urban. A Synthesis of Evaluation Criteria for Software Specifications and Specification Techniques. *International Journal of Software Engineering and Knowledge Engineering*, 12(5):581–599, 2002.

[MD10] R. Matulevičius and M. Dumas. A Comparison of SecureUML and UMLsec for Role-Based Access Control. In *Databases and Information Systems*, pages 171 – 185, 2010.

[VFMP05] R. Villarroel, E. Fernández-Medina, and M. Piattini. Secure information systems development a survey and comparison. *Computers & Security*, 24(4):308 – 321, 2005.

Empirical Validation of Security Methods

Ph.D. Candidate: Katsiaryna Labunets
Advisor: Fabio Massacci

University of Trento, Italy
{surname}@disi.unitn.it

Abstract. Security requirements engineering is an important part of many software projects. Practitioners consider security requirements from the early stages of software development processes, but most of them do not use any formal method for security requirements engineering. According to a recent survey, only about 9% security practitioners implement formal process of elicitation and analysis of security requirements and risks.

However, a number of methods have been recently proposed in academia to support practitioners in collecting and analysing security requirements. Unfortunately, these methods are not widely adopted in practice because there is a lack of empirical evidence that they work. Only few papers in requirements engineering have a solid empirical evidence of efficiency of proposed solutions. So how can we know that security methods work in practice?

In this paper we propose to conduct a series of empirical studies to build a basis that *a*) will provide security practitioners with guidelines for selection of security requirements methods, and *b*) will help methods designer understand how to improve their methods.

1 Introduction

An increasing role of security in software development process is recognized by both industrial professionals [14] and academia members [10]. The security requirements and risk analysis plays a major role in delivery of secure software systems.

A variety of academic security methods like SREP [9] and CORAS [7], Secure Tropos [12] and SI* [6], LINDDUN [3] and misuse cases [15] have been proposed in the last years. However, these methods are not commonly used in industry. Only 9% of security practitioners implement formal process of elicitation and analysis of security requirements and risks [5]. The reason of this can be a lack of empirical study showing the effectiveness of these methods on real cases. In most of the papers in requirement engineering researchers propose a new methodology and shows that it works. This is acknowledged by a recent study of Condori-

Fernandez et al. [2] shows that only 13% of research works in Requirements Engineering relied on case studies[1].

So how can the practitioners decide which method is better for elicitation and analysis of security requirements and risks in their projects? This lack of empirical grounded knowledge on security methods effectiveness in real cases blocks a wide deployment of academic methods in industrial projects. Indeed, disregarding of validation activities is a drawback for both practitioners and methods designers. Practitioners do not know which methods to apply because designers of methods do not provide information about the effectiveness and usefulness of the methods in real cases. Methods designers do not know whether the methods are efficient in practice or not because there is no experience in practical application of the methods.

The main objective of our research is to investigate the effectiveness of security requirements and risk analysis methods and *the reasons* of their effectiveness through a series of empirical studies. The second objective is to build an empirical basis that *a*) can provide security practitioners with guidelines for selection of security requirements methods, and *b*) will help methods designers to understand how to improve their methods.

There is a number of empirical studies are dealing with requirements engineering. Morandini et al. [11] present qualitative study of requirements comprehension. They compare Tropos and Tropos4AS requirements methods. Opdahl et al. [13] carried out a pair of controlled experiments to compare two methods for security threats identification: misuse cases and attack trees. Asnar et al. [1] presents their experience in modeling and analysis of requirements in practice. They applied Secure Tropos method in air transport management system.

The expected contribution of our work to the field of Engineering Secure Software and Systems is practical guidelines for selecting security requirements and risk assessment methods.

The rest of the paper is organized as follows: Section 2 states the proposed research directions. The research methodology and research plan are presented in Section 3. The ongoing and future work are discussed in Section 4.

2 Research Objectives

There is a number of methods for elicitation and analysis of security requirements. Usually methods designers propose a method and claim that the method works. To show that the method works typically method designer apply the method to a real case. This kind of experimentation cannot be accepted as a solid evidence that the new method works in practice.

Without experimentation how do practitioners choose methods for elicitation of security requirements and risks? How do researchers understand ways to improve their method? These two questions lead to the third one: How to empirically validate the effectiveness of a new security method?

[1] Case study is an in-depth investigation of how and why a particular phenomenon happened in real-life conditions

In this work we propose to empirically validate the effectiveness of security requirements methods when they are applied by novices, i.e., users that have no prior knowledge of these methods. In particular, we wand to understand which security methods are effective and which are not, and what are the reasons behind.

The main outcome of our research is to build an empirical ground that *a*) will help security practitioners to select a security requirements method, and *b*) provides methods designers with ideas on how to improve their methods.

Thus, our research work aims to answer the following questions:

- **RQ1** *Do security methods work in practice when applied by novice users?*
- **RQ2** *Why do some methods work? Why others don't?*

We want to answer the above research questions by conducting a series of empirical studies using a mix-method methodology combining research approaches from qualitative research (e.g. grounded theory) and quantitative research (e.g. controlled experiments).

3 Research Methodology and Research Plan

The research can be organized into 3 main phases: *Design* (step 1), *Execution* (steps 2-3), and *Analysis* (step 4). The decomposition of research plan is shown in Figure 1.

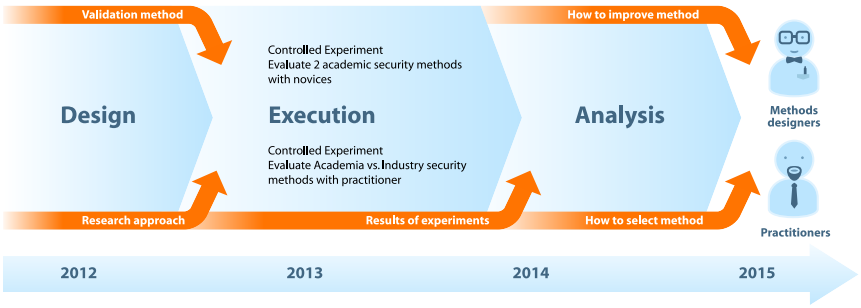


Fig. 1: Research plan.

3.1 Step 1: Selection of Research Methodology

The first step is to select a research methodology for our research. For this purpose we compare the research methodologies that are the most relevant to studies in the field of software engineering. We adopt the Easterbrook et al. [4]

taxonomy of research methods. This taxonomy considers five main classes of research methods: *a*) controlled experiments; *b*) case studies; *c*) survey research; *d*) ethnography; *e*) action research.

We propose to combine qualitative studies (grounded theory) with quantitative research (metrics measurements and statistics). We will apply quantitative methods to evaluate the effectiveness of academic methods for elicitation of security requirements (*RQ1*). Quantitative methods will allow us to collect wide knowledge base for following qualitative studies. For *RQ2* we plan to conduct qualitative studies to find out the underlying reasons of methods effectiveness (*RQ2*). The quantitative methods make it possible to advance hypotheses of security methods and check them experimentally.

We selected controlled experiment as a principal methodology for our research because it allows us to control necessary variables and collect reliable data for evaluation of the effectiveness of security requirements methods. However, we combine controlled experiments with quantitative (metrics measurements) and qualitative methods (focus group interview).

3.2 Step 2: Evaluate the Effectiveness of Academic Security Methods

At this step, in order to answer **RQ1**, we propose to evaluate the effectiveness of academic security requirements methods in case when they are applied by novices. A controlled experiment with master students will allow us to measure the effectiveness of security requirements methods in case when they are applied by users without prior knowledge of the methods.

3.3 Step 3: Academic security method vs. Industrial

The aim of this step it to understand which features make methods effective and working in practice. We propose to evaluate and compare academic versus industrial security methods in order to study which of the studied methods work better and why it happens. For this purpose we propose to conduct a controlled experiment with security practitioners and evaluate the effectiveness of one academic security requirements method (the best one by results of the previous experiment) versus the most used industrial method. The controlled experiment with practitioners and comparison of academic method with industrial one can give us an idea about those details and features that makes a method effective in real cases.

3.4 Step 4: Build Empirical Grounds

At this step we are going to aggregate and analyse the results of our experiments. Basing on these results we plan to develop empirical basis for selection of security methods. We believe this knowledge should help practitioners in understanding which of academic security methods are suitable for the purposes of their projects. At the same time the methods designers may find an idea about what makes their security methods applicable.

4 Ongoing and Future Work

This section presents the ongoing work and supplements the research plan with details of the future work.

The first experiment. Based on the results of eRISE challenges [8] conducted in 2011 and 2012 we selected 2 academic security methods, SREP and CORAS, to be evaluated and compared in our experiment. The last eRISE challenge showed that the SREP method was more appreciated by the participants than CORAS. Every participant that applied SREP method was able to finish all steps and identified a set of security requirements. In contrast, CORAS method showed worse results. The aim of the controlled experiment is to understand the reasons that make SREP better than CORAS. Quantitative research will help us to compare the effectiveness of the methods and find the reasons behind.

We are conducting controlled experiment with master students of Security Engineering course at University of Trento. The master students are divided into two groups. The first group of students uses SREP method while the second group applies CORAS method and vice-versa. During the experiment for each method-case combination we measure a number of metrics.

To evaluate the performance of the methods we take the following metrics:

- *Number of assets.* How many threats per asset does the participant identify for the case?
- *Number of threats.* How many threats does the participant identify for the case?
- *Number of security requirements.* How many security requirements does the participant identify for the case? [2]

To evaluate the perception of the users we take values similar to Opdahl et al. [13]:

- *Perceived usefulness.* How useful does the participant consider the method to be?
- *Perceived easy to use.* How easy to use does the participant consider the method to be?
- *Intention to use.* Does the participant intend to use the method again in the future?

In this experiment we propose to test more than 10 hypotheses that are dealing with correlation between performance and perception metrics, and the participants experience in application of security methods. Here are some examples of hypotheses to test:

[2] We use the notion of security requirements to denote both treatments in CORAS and security requirements in SREP because they both define a way to mitigate a threat.

– **H1** There will be a difference in the number of threats found with CORAS and with SREP within each case.
– **H2** The difference between the numbers of security requirements found with CORAS and with SREP will be correlated with the difference between the participants' preferences for CORAS and for SREP within each case
– **H3** The difference between the number of threats found with CORAS and with SREP will be correlated with participants' knowledge in security

We foreseen that the experiment will show *a*) how users without experience in application of security methods can comply with the method guidelines and *b*) how the first experience in application of one security method impacts on the work with other security methods (i.e., does the previous experience with security methods facilitate the application of new ones?)

eRISE 2013. In 2013 we will be involved in organization of eRISE 2013 challenge. This challenge aims to empirically evaluate security requirements and risk assessment methods. During eRISE we will study how participants apply the methods in practice, and which security methods are more effective and what features make them useful. The eRISE 2013 covers the objectives of our second experiment. We will conduct a controlled experiment with practitioners and compare the best *academic method* by the results of our first experiment and an *industrial method* that is the most used in practice. We expect to collect sufficient and reliable data to understand what the security methods need to become effective.

References

1. Y. Asnar, P. Giorgini, F. Massacci, A. Saidane, R. Bonato, V. Meduri, and V. Ricucci. Secure and dependable patterns in organizations: An empirical approach. In *Proc. of RE '07*, pages 287 –292, 2007.
2. N. Condori-Fernandez, M. Daneva, K. Sikkel, R. Wieringa, O. Dieste, and O. Pastor. A systematic mapping study on empirical evaluation of software requirements specifications techniques. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 502–505. IEEE Computer Society, 2009.
3. M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32, 2011.
4. S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. Selecting empirical methods for software engineering research. In F. Shull, J. Singer, and D. Sjberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer London, 2008.
5. G. Elahi, E. Yu, T. Li, and L. Liu. Security requirements engineering in the wild: A survey of common practices. In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pages 314–319. IEEE, 2011.
6. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 167–176. IEEE, 2005.

7. M. Lund, B. Solhaug, and K. Stølen. *Model-driven risk analysis: the CORAS approach*. Springer, 2010.
8. F. Massacci and F. Paci. How to select a security requirements method? A comparative study with students and practitioners. *Secure IT Systems*, pages 89–104, 2012.
9. D. Mellado, E. Fernández-Medina, and M. Piattini. Applying a security requirements engineering process. *Computer Security–ESORICS 2006*, pages 192–206, 2006.
10. D. Mellado and D. Rosado. An overview of current information systems security challenges and innovations J. UCS Special Issue. *Journal of Universal Computer Science*, 18(12):1598–1607, 2012.
11. M. Morandini, A. Marchetto, and A. Perini. Requirements comprehension: A controlled experiment on conceptual modeling methods. In *Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on*, pages 53–60. IEEE, 2011.
12. H. Mouratidis. Secure software systems engineering: the Secure Tropos approach. *Journal of Software*, 6(3):331–339, 2011.
13. A. L. Opdahl and G. Sindre. Experimental comparison of attack trees and misuse cases for security threat identification. *Inf. Softw. Technol.*, 51(5):916–932, 2009.
14. B. Schneier. The importance of security engineering. *Security Privacy, IEEE*, 10(5):88, Sept.-Oct. 2012.
15. G. Sindre and A. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.