ROYAL INSTITUTE
OF TECHNOLOGY

A framework and theory for cyber security assessments

Teodor Sommestad

2012

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Industrial Information and Control Systems
KTH, Royal Institute of Technology
Stockholm, Sweden

TRITA-EE 2012:043
ISSN 1653-5146
ISRN KTH/ICS/R—12/03—SE
ISBN 978-91-7501-511-8

Stockholm 2012, Universitetsservice US AB

Abstract

Information technology (IT) is critical and valuable to our society. An important type of IT system is Supervisor Control And Data Acquisition (SCADA) systems. These systems are used to control and monitor physical industrial processes like electrical power supply, water supply and railroad transport. Since our society is heavily dependent on these industrial processes we are also dependent on the behavior of our SCADA systems. SCADA systems have become (and continue to be) integrated with other IT systems they are thereby becoming increasingly vulnerable to cyber threats. Decision makers need to assess the security that a SCADA system's architecture offers in order to make informed decisions concerning its appropriateness. However, data collection costs often restrict how much information that can be collected about the SCADA system's architecture and it is difficult for a decision maker to know how important different variables are or what their value mean for the SCADA system's security.

The contribution of this thesis is a modeling framework and a theory to support cyber security vulnerability assessments. It has a particular focus on SCADA systems. The thesis is a composite of six papers. Paper A describes a template stating how probabilistic relational models can be used to connect architecture models with cyber security theory. Papers B through E contribute with theory on operational security. More precisely, they contribute with theory on: discovery of software vulnerabilities (paper B), remote arbitrary code exploits (paper C), intrusion detection (paper D) and denial-of-service attacks (paper E). Paper F describes how the contribution of paper A is combined with the contributions of papers B through E and other operationalized cyber security theory. The result is a decision support tool called the Cyber Security Modeling Language (CySeMoL). This tool produces a vulnerability assessment for a system based on an architecture model of it.

Keywords: cyber security, security assessment, vulnerability assessment, architecture modeling, enterprise architecture.

Sammanfattning

Informationsteknik (IT) är kritiskt och värdefullt för vårt samhälle. En viktig typ av IT-system är de styrsystem som ofta kallas SCADA-system (från engelskans ”Supervisor Control And Data Acquisition”). Dessa system styr och övervakar fysiska industriella processer så som kraftförsörjning, vattenförsörjning och järnvägstransport. Eftersom vårt samhälle är beroende av dessa industriella processer så är vi också beroende av våra SCADA-systems beteende. SCADA-system har blivit (och fortsätter bli) integrerade med andra IT system och blir därmed mer sårbara för cyberhot. Beslutsfattare behöver utvärdera säkerheten som en systemarkitektur erbjuder för att kunna fatta informerade beslut rörande dess lämplighet. Men datainsamlingskostnader begränsar ofta hur mycket information som kan samlas in om ett SCADA-systems arkitektur och det är svårt för en beslutsfattare att veta hur viktiga olika variabler är eller vad deras värden betyder för SCADA-systemets säkerhet.

Bidraget i denna avhandling är ett modelleringsramverk och en teori för att stödja cybersäkerhetsutvärderingar. Det har ett särskilt focus på SCADA-system. Avhandlingen är av sammanläggningstyp och består av sex artiklar. Artikel A beskriver en mall för hur probabilistiska relationsmodeller kan användas för att koppla samman cybersäkerhetsteori med arkitekturmodeller. Artikel B till E bidrar med teori inom operationell säkerhet. Mer exakt, de bidrar med teori angående: upptäckt av mjukvarusårbarheter (artikel B), fjärrexekvering av godtycklig kod (artikel C), intrångsdetektering (artikel D) och attacker mot tillgänglighet (artikel E). Artikel F beskriver hur bidraget i artikel A kombineras med bidragen i artikel B till E och annan operationell cybersäkerhetsteori. Resultatet är ett beslutsstödsverktyg kallat Cyber Security Modeling Language (CySeMoL). Beslutsstödsverktyget producerar sårbarhetsutvärdering för ett system baserat på en arkitekturmodell av det.

Nyckelord: cybersäkerhet, säkerhetsvärdering, sårbarhetsvärdering, arkitekturmodellering.

Preface

When my research on this topic began in early 2007 the American cyber security regulation NERC CIP was a buzzword, and electrical power utilities in my surroundings began to become aware of the cyber security issues related to their SCADA systems. During my first years of working with cyber security of SCADA systems I often ended up in discussions concerning the relevance of the topic with those who owned the problem, i.e., asset owners and SCADA system suppliers. These discussions were on aspects such as: if there were a threat at all, why cyber attacks would be used instead of dynamite and what a cyber attack against a SCADA system possibly could accomplish. Now, at the finalization of this thesis, the computer worm (or "cyber weapon") Stuxnet still gets headlines in prominent magazines and papers, even though it was discovered more than two years ago. During the past two years, my discussions with problem-owners have focused on finding and describing solutions, and not on debating whether there is a problem worth considering. I sincerely hope that this thesis, along with the other outputs produced during my PhD studies (e.g., the tool supporting applications of these theories), will help to make our SCADA systems more secure.

As is customary in the Swedish system, this thesis is divided into two parts. The first part summarizes and gives an overview of the second part. In the second part the actual contributions are presented. The actual contributions are six of the papers produced during my doctoral studies. These six papers all contribute to the problem of assessing the cyber security of a system. The first paper presents a template which can be used to express security theory so that it can be directly applied on a system model. Papers two through five present theories on the topic and paper six presents a software tool that combines the formalism and the theory in order to support cyber security vulnerability assessments.

It is difficult to produce an exhaustive list of all those who have helped, contributed, and supported me during this journey. In addition to my colleagues at the department and paper co-

authors (especially Hannes Holm), I would like to thank associate professor Mathias Ekstedt, professor Pontus Johnson, and professor Lars Nordström for their guidance. I would also like to give a special thanks to Judith Westerlund and my wife Caroline for their support and encouragement. Finally, I would like to thank all the security experts who have contributed to my research projects.

[illegible]

Teodor Sommestad

List of included papers

Paper A: T. Sommestad, M. Ekstedt, and P. Johnson, "A probabilistic relational model for security risk analysis," *Computers & Security*, vol. 29, no. 6, pp. 659-679, 2010.

Paper B: T. Sommestad, H. Holm, and M. Ekstedt, "Effort estimates for vulnerability discovery projects," in *Proceedings of the 45th Hawaii International Conference on System Sciences*, pp. 5564-5573, 2012.

Paper C: T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of remote arbitrary code execution attacks," *Information Management & Computer Security*, vol. 20, no. 2, pp. 107-122, 2012.

Paper D: T. Sommestad, H. Holm, M. Ekstedt, and N. Honeth, "Quantifying the effectiveness of intrusion detection systems in operation through domain experts," Submitted.

Paper E: T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of Denial-of-Service attacks," in *Proceedings of IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 21-28, 2011.

Paper F: T. Sommestad, M. Ekstedt, and H. Holm, "The Cyber Security Modeling Language: A Tool for Vulnerability Assessments of Enterprise System Architectures," IEEE Systems Journal, Accepted for publication.

Contribution in the included papers: In all papers (A-F) Teodor Sommestad has been the leading researcher and primary author. In paper A Mathias Ekstedt and Pontus Johnson assisted with problem definition and authoring. In papers B-E Hannes Holm had an active role in the research and did approximately a third of the authoring. Mathias Ekstedt contributed to papers B-E in terms of authoring and advice on method selection and survey design. Nicholas Honeth contributed to the authoring and survey design in paper D. Mathias Ekstedt contributed to paper F in terms of the overall idea and authoring. Hannes Holm contributed to paper F with a case study.

Publications not included in the thesis

Publication I: P. Johnson, E. Johansson, T. Sommestad, and J. Ullberg, "A tool for enterprise architecture analysis," in *Proceedings of Enterprise Distributed Object Computing Conference*, 2007, pp. 142–142.

Publication II: M. Ekstedt, P. Johnson, M. Gammelgård, T. Sommestad, and P. Gustafsson, "Setting the Business Goals," in *Enterprise Architcture: models and analyses for information systems decision making*, Sweden: Studentlitteratur AB, 2007.

Publication III: P. Johnson, M. Ekstedt, R. Lagerström, and T. Sommestad, "Introduction," in *Enterprise Architcture: models and analyses for information systems decision making*, Sweden: Studentlitteratur AB, 2007.

Publication IV: U. Franke, T. Sommestad, M. Ekstedt, and P. Johnson, "Defense Graphs and Enterprise Architecture for Information Assurance Analysis," in *Proceedings of the 26th Army Science Conference*, 2008.

Publication V: T. Sommestad, M. Ekstedt, and P. Johnson, "Combining Defense Graphs and Enterprise Architecture Models for Security Analysis," in *Proceedings of 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, 2008.

Publication VI: E. Johansson, T. Sommestad, and M. Ekstedt, "Security Isssues For SCADA Systems within Power Distribution," in *Proceedings of Nordic Distribution and Asset Management Conference (NORDAC)*, 2008.

Publication VII: Y. Xiaofeng, T. Sommestad, C. Fung, and P. C. K. Hung, "Emergency Response Framework for Aviation XML Services on MANET," in *Proceedings of The IEEE International Conference on Web Services (ICWS)*, 2008.

Publication VIII: U. Franke, J. Ullberg, T. Sommestad, R. Lagerström, and P. Johnson, "Decision support oriented

Enterprise Architecture metamodel management using classification trees," in *2009 13th Enterprise Distributed Object Computing Conference Workshops*, 2009.

Publication IX: E. Johansson, T. Sommestad, and M. Ekstedt, "Issues of Cyber Security In Scada-Systems-on the Importance of Awareness," in *Proceedings of the 20th International Conference on Electricity Distribution (CIRED)*, 2009.

Publication X: P. Närman, T. Sommestad, S. Sandgren, and M. Ekstedt, "A framework for assessing the cost of IT investments," in *PICMET 2009 Proceedings*, 2009.

Publication XI: T. Sommestad, M. Ekstedt, and L. Nordstrom, "Modeling Security of Power Communication Systems Using Defense Graphs and Influence Diagrams," *IEEE Transactions on Power Delivery*, vol. 24, no. 4, pp. 1801-1808, 2009.

Publication XII: S. Buckl, U. Franke, O. Holschke, F. Matthes, C.M. Schweda, T. Sommestad, and J. Ullberg, "A Pattern-based Approach to Quantitative Enterprise Architecture Analysis," in *Proceedings of 15th Americas Conference on Information Systems (AMCIS)*, 2009.

Publication XIII: M. Ekstedt, U. Franke, P. Johnson, R. Lagerström, T. Sommestad, J. Ullberg, and M. Buschle, "A Tool for Enterprise Architecture Analysis of Maintainability," in *Proceedings of the 2009 European Conference on Software Maintenance and Reengineering*, 2009.

Publication XIV: W. R. Flores, T. Sommestad, P. Johnson, and M. Simonsson, "Indicators predicting similarities in maturity between processes: An empirical Analysis with 35 European organizations," in *Proceedings of 1st Annual Pre-ICIS Workshop on Accounting Information Systems*, 2009.

Publication XV: T. Sommestad, M. Ekstedt, and P. Johnson, "Cyber Security Risks Assessment with Bayesian Defense Graphs and Architectural Models," in *Proceedings of Hawaii International Conference on System Sciences*, 2009.

Publication XVI: M. Ekstedt and T. Sommestad, "Enterprise Architecture Models for Cyber Security Analysis," in *Proceedings of IEEE PES Power Systems Conference & Exhibition (PSCE)*, 2009.

Publication XVII: M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A Tool for Enterprise Architecture Analysis using the PRM formalism," in *CAiSE2010 Forum PostProceedings*, 2010.

Publication XVIII: M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A Tool for Enterprise Architecture Analysis using the PRM formalism," in *Proceedings of CAiSE Forum 2010*, 2010.

Publication XIX: T. Sommestad, G. Björkman, M. Ekstedt, and L. Nordström, "Information system architectures in electrical distribution utilities," in *Proceedings of NORDAC*, 2010.

Publication XX: G. Björkman, T. Sommestad, M. Ekstedt, H. Hadeli, Z. Kun, and M. Chenine, *SCADA system architectures*. Stockholm, Sweden: Report of The VIKING project, 2010.

Publication XXI: F. Löf, J. Stomberg, T. Sommestad, M. Ekstedt, J. Hallberg, and J. Bengtsson, "An Approach to Network Security Assessment based on Probabilistic Relational Models," in *First Workshop on Secure Control Systems (SCS-1)*, 2010.

Publication XXII: T. Sommestad, M. Ekstedt, and L. Nordström, "A case study applying the Cyber Security Modeling Language," in *Proceeding of CIGRE (International Council on Large Electric Systems)*, 2010.

Publication XXIII: T. Sommestad, G. Ericsson, and J. Nordlander, "SCADA System Cyber Security – A Comparison of Standards," in *Proceedings of IEEE PES General Meeting*, 2010.

Publication XXIV: T. Sommestad and J. Lillieskold, "Development of an effort estimation model – a case study on delivery projects at a leading IT provider within the electric utility industry," *International Journal of Services Technology and Management*, vol. 13, no. 1/2, p. 152, 2010.

Publication XXV: H. Holm, T. Sommestad, J. Almroth, M. Persson, "A quantitative evaluation of vulnerability scanning", *Information Management & Computer Security*, vol. 19, no. 4, pp. 231-247, 2011.

Publication XXVI: T. Sommestad, *Exploiting network configuration mistakes: practitioners self-assessed success rate*. Stockholm, Sweden: TRITA-EE 2011:069, 2011.

Publication XXVII: T. Sommestad, H. Holm, and M. Ekstedt, *Threats and vulnerabilities, final report*. Stockholm, Sweden: Report of The VIKING project, 2011.

Publication XXVIII: H. Holm, T. Sommestad, and M. Ekstedt *Vulnerability assessment of SCADA systems*. Stockholm, Sweden: Report of The VIKING project, 2011

Publication XXIX: T. Sommestad, *Password authentication attacks: a survey of attacks and when they will succeed*. Stockholm, Sweden: TRITA-EE 2011:067, 2011.

Publication XXX: H. Holm, T. Sommestad, U. Franke, M. Ekstedt, "Expert Assessment on the Probability of Successful Remote Code Execution Attacks," in *Proceedings of WOSIS 2011 - Proceedings of the 8th International Workshop on Security in Information Systems, In conjunction with ICEIS 2011*, Beijing, China, 49-58, 2011

Publication XXXI: T. Sommestad and J. Hallberg, "Cyber security exercises as a platform for cyber security experiments," in *TAMSEC*, 2011, p. 33.

Publication XXXII: W. Flores, T. Sommestad, and H. Holm, "Assessing Future Value of Investments in Security-Related IT Governance Control Objectives – Surveying IT Professionals," *The Electronic Journal of Information Systems Evaluation*, vol. 14, no. 2, pp. 216-227, 2011.

Publication XXXIII: H. Holm, T. Sommestad, U. Franke, and M. Ekstedt, "Success rate of remote code execution attacks – expert assessments and observations," *Journal of Universal Computer Science*, vol. 18, no. 6, pp. 732-749, 2012.

Publication XXXIV: T. Sommestad and A. Hunstad, "Intrusion detection and the role of the system administrator," in *Proceedings of International Symposium on Human Aspects of Information Security & Assurance*, 2012.

Publication XXXV: T. Sommestad and J. Hallberg, "Cyber security exercises and competitions as a platform for cyber security experiments," in *Proceedings of the 17th Nordic Conference on Secure IT Systems*, 2012.

Table of contents

Part one: Summary.......1

1 Introduction.......2

1.1 Outline of the thesis.......2

1.2 Background.......2

1.3 Objectives.......5

2 Related works.......6

2.1 Metrication frameworks and methods.......7

2.2 Operationalized cyber security theory.......9

2.3 Operationalized cyber security assessment methods.......12

3 Result and contribution.......13

3.1 Means of representation.......14

3.2 Constructs.......17

3.3 Statements of relationship.......18

3.4 Scope.......20

3.5 Causal explanations.......22

3.6 Testable propositions.......22

3.7 Prescriptive statements.......25

4 Research design.......26

4.1 Framework and formalism.......26

4.2 Qualitative theory.......27

4.3 Quantitative theory.......28

4.4 Validation.......30

5 References.......32

Part two: Papers 43

Paper A: A probabilistic relational model for security risk analysis 44

Paper B: Effort estimates for vulnerability discovery projects...............93

Paper C: Estimates of success rates of remote arbitrary code execution attacks 117

Paper D: Quantifying the effectiveness of intrusion detection systems in operation through domain experts...141

Paper E: Estimates of success rates of denial-of-service attacks..........176

Paper F: The Cyber Security Modeling Language: A Tool for Vulnerability Assessments of Enterprise System Architectures 200

Part one: Summary

1 Introduction

This introduction describes the thesis' outline, the background of the research and the objectives of the research.

1.1 Outline of the thesis

This thesis is divided into two parts. Part one (this part) presents an overview and part two presents a summary to the actual contribution.

The remainder of section 1 in this part of the thesis gives a description of the background and objectives of the research. Section 2 describes related works and relates this to the contribution of this thesis. Section 3 summarizes the contribution of this thesis by presenting properties of the theory presented in it. Section 4 describes the research design.

The second part of the thesis contains six papers labeled papers A through F. Two of these papers have been published in the proceedings of international conferences, three have been accepted or published in international journals, and one is currently under review for publication at an international journal. The papers contain the same content as when they were published/accepted/submitted, only their typesetting has been changed.

1.2 Background

Information technology (IT) is critical and valuable to our society. IT systems support business processes by storing, processing, and communicating critical and sensitive business data. In addition, IT systems are often used to control and monitor physical industrial processes. For example, our electrical power supply, water supply and railroads are controlled by IT systems. These "controlling" systems have many names. In this thesis they are referred to as SCADA (Supervisory Control And Data Acquisition) systems, or occasionally, as industrial control systems. They are complex real-time systems that include components like databases, application servers, web interfaces, human machine interfaces, dedicated communication equipment, process control logic, and numerous sensors and actuators that measure and control the state of the industrial process. In many industrial processes (e.g., electrical power transmission) these components are also distributed over a large geographical area. SCADA systems can be seen as the nervous system of industrial processes [1] and since our society is heavily dependent on the industrial processes that SCADA systems manage, we are also dependent on the behavior of our SCADA systems.

Over the last two decades our SCADA systems and their environments have changed. They used to be built on proprietary and specialized protocols and platforms [2]. Today, however, SCADA systems operate on top of common and widely used operating systems (e.g., Windows XP) and use protocols that are standardized and publicly available (e.g., IEC 60870-5-104). These changes have altered the threat environment for SCADA systems.

The move to more well-known and open solutions lowers the threshold for attackers who seek to exploit vulnerabilities in these SCADA systems. Vulnerabilities are regularly found in the software components used in SCADA systems (e.g., the operating systems) and instructions that can be used to exploit these vulnerabilities are often made available in the public domain. The increased openness also lowers the thresholds for attacks targeting special-purpose SCADA components, e.g., programmable logic controllers (PLCs). Today there is an interest in the vulnerabilities they have and there is information available in the public domain about their design and internal components. In fact, it is even possible to buy a subscription to exploit code specifically targeting SCADA systems' components (see for example [3]). In other words, a successful cyber attack against a SCADA system today does not require the SCADA-expertise that was required prior to the move to more open, standardized and common components.

In parallel with the move to more common and widely known solutions, SCADA systems have moved from being isolated and standalone to be interwoven in the larger IT environment of enterprises. Process data collected by SCADA systems, production plans, and facility drawings are often exchanged over enterprises' computer networks [4]. It is also common to allow users to remotely connect to operator interfaces, for instance, so that process-operators can connect remotely when they are on standby duty and so that suppliers are able to perform maintenance remotely [4].

The increased integration with more administrative enterprise systems has also contributed to a changed threat environment. Administrative systems are, with few exceptions, connected (directly or indirectly) to the internet. Hence, the possibility for administrative systems to exchange data with SCADA systems is also a possibility for attackers or malware to come in contact with these systems and exploit their vulnerabilities, without physical proximity.

The lowered threshold to find and use SCADA-related vulnerabilities and tighter integration with enterprise systems are two cyber security problems that add to the volume of cyber security issues related to architecture and configuration of the actual SCADA systems [5–7]. Historically, SCADA systems were built to be reliable and available, but not to be secure against attacks with a malicious intent.

SCADA systems are thus critical assets, have exploitable vulnerabilities, and are interwoven into the enterprise architectures. Decision makers who wish to manage their cyber security need to be able to assess the vulnerabilities associated with different solution architectures. However, assessing the cyber security of an enterprise environment is difficult. The budget allocated for cyber security assessments is usually limited. This prohibits assessments from covering and investigating all factors that could be of importance. The set of variables that should be investigated, and how important they are, is also hazy and partly unknown. For instance, guidelines such as [8–11] do not prioritize their cyber security recommendations. Such prioritizations are also difficult to do in a generic guideline since the importance of many variables are contingent on the systems architecture and environment and guidelines are limited to one or few typical architectures. Variables are also dependent on each other. An attack against a SCADA system may be performed in a number of ways and can involve a series of steps where different vulnerabilities are exploited. Thus, some combinations of vulnerabilities can make an attack easy, but a slightly different combination may make attacks extremely difficult. Thus, informed decisions require an analysis of the vulnerabilities associated with different architectural scenarios, and at the same time, an analysis of how these vulnerabilities relate to each other.

These problems are not unique for SCADA systems. Many administrative IT systems also have complex environments; administrative IT systems often need to be analyzed on a high level of abstraction; the importance of different variables is hazy also for administrative IT systems. Like the administrative environment, the SCADA environment consists of software, hardware, humans, and management processes. And as described above, there is a substantial overlap between the components which are used in both environments today. However, there is a difference in what needs to be protected in these environments. Security is often thought of as a triage of confidentiality, integrity and availability. For SCADA systems, integrity and availability of functionality are crucial, but confidentiality of business data is not [9]. Because of this, cyber security assessments of SCADA systems have a different focus than for many other systems. The importance of availability and integrity has also other implications. For instance, because of the consequence of a potential malfunction, it is recommended that SCADA systems should not be updated before extensive testing, and network based vulnerability scanners should be used with care in SCADA environments [9].

1.3 Objectives

The overall aim of this research is to develop support for those conducting cyber security assessments. More precisely, the objective is to: *Develop a tool that makes cyber security theory easy to use for decision makers*. To reach this objective the two sub-objectives were identified:

(1) *Define a formalism that makes it possible to apply a cyber security theory on system architecture specifications* and
(2) *Compile and develop cyber security theory that is relevant for decision makers in the SCADA domain.*

The purpose of this research is thus to help decision makers to assess the cyber security of IT systems with different architectures. Help is needed to assesses both existing systems "as-is" and potential future "to-be" systems. Focus is on supporting decision makers in the SCADA domain. As presented above (cf. section 1.2) such support must tackle practical issues. First, cyber security assessment cannot be overly costly to perform, viz. all details concerning the SCADA system's architecture and configuration cannot be investigated. Second, the theory on what makes a system secure is, is not always clear (especially when details about the system are missing) and in approximations are necessary. Both these practical issues make assessments uncertain and to support a decision maker, trade-offs are needed with respect to accuracy. The aim is to produce a reasonable tradeoff between accuracy and the cost of collecting system specific data while communicating the uncertainty of the result.

2 Related works

The contribution of this thesis follows ideas of the management approach called enterprise architecture. Enterprise architecture is an approach for holistic management of information systems where diagrammatic descriptions of systems and their environment are central. A number of established enterprise architecture frameworks exist, including: The Open Group Architecture Framework [12], the Ministry of Defence Architecture Framework [13] and the Department of Defense Architecture Framework [14]. The research presented in this thesis follows the ideas presented in [15], [16] concerning enterprise architecture modeling and decision making. The overall idea is that the concepts represented in (enterprise) architecture models should be there because they, according to theory, are needed to answer questions of interest to the decision maker that uses the architecture for some specific purpose.

This thesis focuses on questions related to cyber security and how to answer those questions with the support of architectural models of systems. While established some enterprise architecture frameworks do address security explicitly, the analysis support they offer is sparse. For instance, in the process suggested by The Open Group Architecture Framework [12] includes steps where one should "Identify potential/likely avenues of attack" and "Determine what can go wrong?", however, it is up to the user of the method (the architect) to do so. Similarly, the support offered by the Ministry of Defence Architecture Framework is to document the result of a security assessment, not to support the analysis required to do it. As described in [17]: "the aim of this guidance for representing security considerations is to enable sufficient information to be recorded for interested parties".

The thesis describes a framework for connecting system architecture models to cyber security assessment (paper A), theory to aid such assessments (papers B-E) and the combination of these into a model that can be described as an expert system (paper F). The three sections below are intended to provide an overview of related work in the directions of the included papers. More elaborate descriptions can be found in the corresponding papers.

Section 2.1 describes methods and models for cyber security assessments. These methods and models require operationalized cyber security theory or system-specific cyber security data (e.g., mean-time to compromise data) to be able to operate. Work on operationalized theory is described in section 2.2. Section 2.3 describes methods and tools that use operationalized cyber security theory to help decision makers assess cyber security.

2.1 Metrication frameworks and methods

A number of ideas can be found on how cyber security should be assessed. Some ideas concern how security measurements should be defined and operationalized. Examples include the ISO/IEC standard 27000-4 [18] and NIST's security metric guide [19]. These publications describe how an organization should develop and maintain a measurement program, but do not define the actual measurements that should be made or what different measurement values mean in terms of security. In addition to these there are general qualitative models that describe variables (or concepts) in the security domain and how these concepts relate to each other. CORAS contains a metamodel over the security field to support assessments made using the CORAS method [20], Common Criteria has a conceptual model over variables (or concepts) a security assessment needs to consider [21] and several similar qualitative models are available. For instance, [22–29] are generic alternatives and [30], [31] are alternatives with a particular focus on SCADA systems that control energy systems. These methods, security metamodels, conceptual models and technical reference models can support cyber security assessments and be used to define operational cyber security metrics. However, they require a substantial mental effort from their user – the user must identify what to measure and how important this is for the IT system's cyber security.

To ease this burden, articles published in scientific forums on security measurement often describe methods to combine security-variables into one metric. Broadly speaking, they define which cyber security variables that should be operationalized and how they should be combined. Examples include: attack trees [32], threat trees[33], defense trees [34], attack and protection trees [35], Boolean Logic Driven Markov Processes [36], the CORAS method [20], XMASS [37], ISRAM [38], NIST's risk assessment framework [39], the economic framework given in [40] and Secure Tropos [41]. Some metrication methods have also been proposed specifically for SCADA systems (e.g., [42–44]).

These metrication methods describe how their variables should be combined to produce a meaningful result. They can thus help to combine cyber security values of single systems to a value for a system-of-systems (e.g., the expected monetary loss next year due to attacks). However, they all require that cyber security theory is supplied by the user. In some cases both qualitative and quantitative theory is needed. For instance, the actual trees together with their attack success probabilities are needed for defense trees [34] and the attacker's process model together with time-to-compromise data is required for Boolean Logic Driven Markov Processes [36]. In some metrication methods the qualitative theory is complete and the user is only required to supply the system architecture and quantitative theory. One example is the model of Breu et al. [45] which requires threat realization probabilities, but describes which threat realization probabilities that are needed and how they should be combined for the modeled enterprise system. Another example is XMASS [37], which among other things requires that the modeler can acquire or specify "security profiles" for entities. With these security profiles a user can calculate an ordinal "security value" (between 0 and 100) for the components in the system.

Paper A describes a framework that can be used to tie security theory to architecture metamodels. Just as the model of Breu et al. [45] and XMASS [37] it can be used to infer the security properties that needs to be quantified from the system architecture. Like XMASS the framework described in paper A makes it possible to store security theory so that security can be assessed without employing security expertise to quantify security properties. Unlike XMASS the framework in paper A stores theory expressed in with concepts directly corresponding to states and events in the real world (e.g., attacks' success given use of certain countermeasures), and the framework produces output that are expressed in tangible units (e.g., expected monetary losses).

2.2 Operationalized cyber security theory

The metrication methods described in section 2.1 needs to be complemented with quantitative cyber security theory to be of practical use. This theory can be supplied together with the metrication method or supplied by the user of the method. The accuracy of the result when the method is applied will of course be contingent on the accuracy of the theory with which it is used. Many prominent research results have been produced on operational cyber security. Some are also specifically addressing the cyber security of SCADA systems (e.g., the demonstrations, assessments and tests described in [46–50]). Unfortunately only a small portion of these could be used in analyses of the types dealt with in this thesis. This section aims at giving an overview of available theory that has been used as a basis for this research and to point to gaps which are filled by papers B-E. More elaborate descriptions of studies related to the contributions in papers B-E can be found in the papers included in part two of this thesis.

Some areas of cyber security have an intrinsic quantitative element which makes metrication and estimation of the required effort to accomplish an attack straightforward [51]. In particular, established methods are available for assessing the strength of cryptographic methods and authentication methods (e.g., password authentication) under well specified conditions [51]. In other fields, empirical investigations have approximated the probability that the attacker would succeed with different attacks on the level of abstraction manageable in an enterprise security assessment (considering the cost of collecting data). For example, studies on social engineering attacks have produced success frequencies under different conditions [52–55]. Other studies have assessed the frequency of configuration mistakes in enterprises' systems and how difficult such mistakes are to exploit [56], [57]. Results described in these papers make up a subset of the theory used in the model of paper F.

With respect to software vulnerabilities there is empirical data available concerning public disclosed software vulnerabilities in databases like [58], [59]. In these, and in databases like [60], it is also possible to identify the vulnerabilities for which exploit code is publicly available. Models have been developed to predict how many cyber security vulnerabilities that will be publicly disclosed for a product [61–64]. For instance, the number of vulnerabilities found in a software product has been found to correlate to the number of user-months the product has accumulated and the time it has been on the market [62]. The effectiveness of different procedures for deploying security patches has also been assessed [65]. When it comes to development of new exploits it is reasonable to assume that this is a straightforward task for a professional penetration tester when patch information is available for the vulnerability. For instance, it is demonstrated in [66] that exploit development can be automated for selected classes of vulnerabilities under those circumstances. However, to predict how difficult it would be for an attacker to find a zero-day vulnerability (i.e., a vulnerability discovered by someone, but which is still unknown to the public and the system owner) in a software product and develop an exploit for it is more difficult. In [67] it is estimated how many zero day vulnerabilities there have been at different points in time during recent years. However, since data on the effort invested in the discovery projects identifying these vulnerabilities (or those projects that failed to identify a software vulnerability) is unavailable [61], it is difficult to deduce the required effort for finding a new vulnerability from the archival records available. Paper B contributes to this with effort estimates for discovery projects undertaken given different conditions.

Several studies have investigated the exploitation of software vulnerabilities, in particular the type of exploitation where a remote attacker obtains control of the vulnerable system. In [68–82] attacks and defenses are described. While these publications describe countermeasures and attacks they mitigate, no study has been found that states how common different conditions and attack forms are, i.e., how often an intelligent attacker will or can employ each of the attack forms studied. Because of this, these studies could not be applied directly to this work. Paper C contributes to this with success rates under different conditions.

Intrusion detection systems monitor systems and aim at identifying attacks made against them. A number of empirical studies have been performed on the probability of attacks being detected and false alarms being produced by these systems (e.g. [83], [84]) and on the impact of different parameters' impact (e.g. [85–87]). However, testing intrusion detection systems in a way that makes the result generalizable to real systems is difficult [88–91]. Studies on intrusion detection systems are also technical and focus on the property of the system alone. In practice, however, it is a tool used by an administrator who monitors its output [92–95] and judges if the alarms are worth reacting upon. A first attempt to assess detection rates when administrators are monitoring the output of the intrusion detection system is described in [96]. While the result of [96] clearly shows the importance of considering system administrators, it is too narrow to offer generic data on intrusion detection systems' effectiveness. Paper D contributes to this with broad and general estimates on how an administrator using an intrusion detection system will perform given different conditions.

Work has also been performed on the denial of service attacks. Examples of experiments, observations and simulations on denial of service attacks and related countermeasures can be found in [97–103]. However, since these studies are made under different assumptions it is difficult to generalize from their results and translate them into a real-world context. Broader reviews in the denial-of-service field [104–108] are also of a qualitative nature. Paper E makes a quantitative contribution in this field and describes approximate success rates under various conditions.

2.3 Operationalized cyber security assessment methods

A number of research efforts prior to the one presented in this thesis operationalize a security assessment method so that decision makers only need to describe their systems in order to obtain the assessment of their enterprise architecture. In other words, there are other assessment methods where the user only needs to input information about the system architecture (and not operationalized security theory). Instead of requiring theory from the user, these assessment methods assign values for security properties (such as time-to-compromise or attack success probability) for the system architecture based on a generic theory.

Research efforts along these lines have in recent years focused on methods that use attack graphs. These methods aim at resolving which attacks can be made against a system architecture. Since potential attacks are the source of cyber security risk, these methods match decision making processes concerning cyber security. The approach were threats and attacks are modeled could be compared to methods that check compliance to a set of standardized security requirements for SCADA systems (e.g., [109], [110]) instead of indicating the vulnerabilities that different solutions have.

Methods based on attack graphs are based on a model over the system architecture and a database of exploits or security vulnerabilities [111], [112]. With this data, an algorithm calculates privileges and network states that can be reached by an attacker starting from a certain position [111]. Since the early variants of attack graphs (like [113], [114]) several tools have been developed with different solutions to the problem. Differences can be seen both in terms of the data they require as input and the output they produce when they are applied. The most mature tools described in the literature are: NetSPA [115], [116], the TVA-tool [117–119] and MulVAL [120].

The operationalized security assessment method presented in this thesis is called CySeMoL (Cyber Security Modeling Language) and is described in paper F. Its conceptual model is similar to that of attack graphs, and like attack graphs it instantiates ways that an attacker can compromise the modeled system. The abstraction level of CySeMoL's analysis is higher than the abstraction level used in attack graph methods like NetSPA, TVA-tool and MulVAL. In particular, CySeMoL does not model individual instance of software vulnerabilities or individual exploits. On the other hand CySeMoL includes more types of entities in the analysis. For example, CySeMoL includes human users and management processes in the analysis.

CySeMoL proposes solutions to some issues with implemented attack graph methods. In particular:

- Unlike NetSPA, CySeMoL does not assume that all vulnerabilities are exploitable on all machines, regardless of configuration.
- Unlike MulVAL, CySeMoL gives arguments for the validity of quantitative data on how difficult it is to exploit a vulnerability.
- Unlike MulVAL and NetSPA, CySeMoL does not rely on the output of vulnerability scanners (which miss many vulnerabilities [121]) to be practically usable.
- Unlike TVA tool, CySeMoL does not require that the user of the model enters exactly which exploits the attacker can use.
- Unlike MulVAL and TVAtool, CySeMoL can assess attacks against client software.
- Unlike these three tools, CySeMoL covers more attack types than exploitation of software vulnerabilities.

The relationship to other operationalized security assessments methods are also described in paper F.

3 Result and contribution

The primary result of this research is a probabilistic relational model containing cyber security theory. This probabilistic relational model and the theory contained in it are henceforth referred to collectively as CySeMoL (Cyber Security Modeling Language). CySeMoL describes how attack steps and countermeasures relate to each other and how they can be used to assess the cyber security of an IT system architecture.

To use CySeMoL, the user supplies an object model complying with CySeMoL's metamodel, states the initial privilege of the attacker and states which attack step the attacker will try to reach (i.e., where the attack will end). With this input CySeMoL can suggest paths the attacker would take and estimate the probability of the attacker succeeding, given that he/she has tried. CySeMoL is thus a theory developed to support cyber security vulnerability assessment. Below, CySeMoL is described using the seven structural components of theories outlined in [122]:

- means of representation
- constructs
- statements of relationships
- scope
- causal explanations
- testable propositions
- prescriptive statements

Each of these theory components is described in a separate subsection below.

3.1 Means of representation

A theory needs to be represented physically in some way [122]. The theory in this thesis is represented through a probabilistic relational model. More specifically, it is represented through a probabilistic relational model complying with the template described in paper A.

A probabilistic relational model (PRM) [123] specifies how a Bayesian network [124] should be constructed from an object model (instance model). In other words, it states how a Bayesian network should be created from a model that instantiates a class diagram (metamodel), such as the one of UML (Unified Modeling Language) [125]. A Bayesian network (sometimes called "causal network" [124]) is a graphical representation of probabilistic dependencies between variables [126]. Hence, a PRM can codify how probabilistic dependencies between objects are contingent on the objects' relationships to each other. As succinctly expressed in [123], PRMs "are to Bayesian networks as relational logic is to propositional logic".

In a PRM the classes can have attributes and reference slots. The attributes are random variables with discrete states; the reference slots point to other classes to state which relationships the class has with other classes. Attributes in the PRM are associated with a set of parents. The parents of an attribute *A* are attributes in the object model which *A*'s value depends upon. The association to an attribute's parents can be used to express qualitative theory. For instance, in Figure 1, attribute *A1* of class *C1* depends on attribute *A2* of class *C2* if objects of these classes are related to each other with reference slot *R1*. How an attribute depends on its parents is defined using a conditional probability table. The probabilities *P1* and *P2* in table of Figure 1 state how attribute *C1.A1* (attribute *A1* for objects of class *C1*) is determined by the value of *C1.R1.A2* (attribute *A2* of the object that *R1* points to). Thus, the theory embedded in PRM is quantified through conditional probabilities.

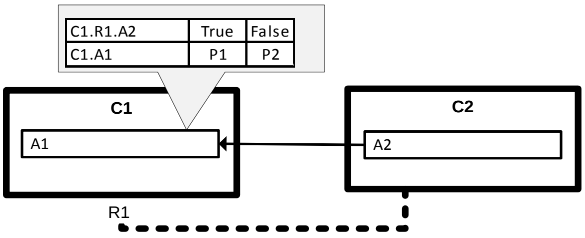


Figure 1. The PRM formalism.

CySeMoL's theory is expressed according to the template depicted in Figure 2. This template is a PRM with abstract classes (i.e., classes that needs to be further refined to be possible to be instantiate in an architecture model). It describes abstract classes that are of relevance to cyber security assessments and describe how the attributes of these classes depend on each other. Among other things, it contains five subclasses to the class *Countermeasure* and details how these influence the cyber security risk. For example, a *PreventiveCountermeasure* influences the probability that an *AttackStep* can be accomplished, while a *ContingencyCountermeasure* influences the loss that would be inflicted on an *Asset* if a *Threat* would be realized.

To summarize, both the qualitative and quantitative parts of the theory are represented through a PRM. An advantage of this means of representation is the possibility of automatically applying the theory on a modeled architecture. A PRM constitutes a formal description for how the value of objects' attributes should be calculated in an object model. Given that a system's architecture is described as an object model, the value of its attributes can be inferred automatically from the theory of the PRM. Such inference can also infer values for attributes which have not been observed, i.e., attributes that do not have a state assigned.

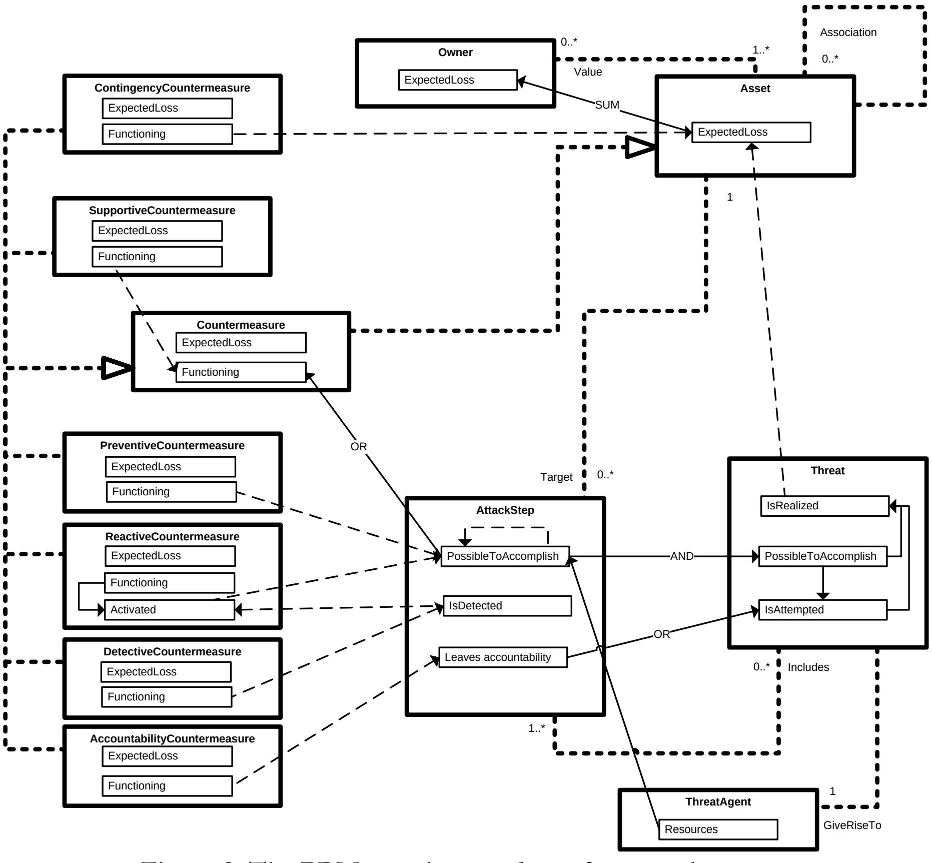


Figure 2. The PRM template used as a framework.

3.2 Constructs

A number of constructs are used in CySeMoL. These constructs are specializations of those in the abstract PRM template (cf. Figure 2). The theory is limited to vulnerability assessments and does not concretize all construct-types in the template. The classes *Asset*, *AttackStep*, *PreventiveCountermeasure*, *DetectivCountermeasure* and *ReactiveCountermeasure* are concretized. One type of *ThreatAgent* is considered, and the *Threat*-class is used but not further concretized.

The theory within CySeMoL is focused on issues concerning SCADA systems. As mentioned in section 1.2, integrity and availability of these systems is the primary concern and confidentiality is not. Also, SCADA systems operate in an environment where certain elements are commonly present and others are not. For instance, bank transactions and mobile phones are not relevant to the typical SCADA system's cyber security. Both the concerns of decision makers and the elements present in SCADA systems' environments have influenced which constructs have been included in CySeMoL.

The metamodel of CySeMoL depicts the constructs of the theory and their relationships to each other. Figure 3 depicts the constructs in terms of classes, attributes and class-relationships (reference slots). Note that this figure is on another level of abstraction than Figure 2, and most attributes in this figure correspond to classes in Figure 2. For example, the attribute *FindHighSeverityVulnerability* in the class *SoftwareInstallation* in Figure 3 is a special type of *AttackStep* (depicted as a class in Figure 2). This is similar to the metamodel layering of UML and the relationship between UML and MOF (Meta Object Facility) [125].

The constructs in CySeMoL have descriptive names. They also have a more elaborate textual definition. For instance, paper C defines and describes a number of the attributes related to arbitrary code exploits. The definitions are intended to be intuitive and accepted in the community. For example, the Common Vulnerability Scoring System's definitions [127] are used in paper C to define properties of attacks and vulnerabilities.

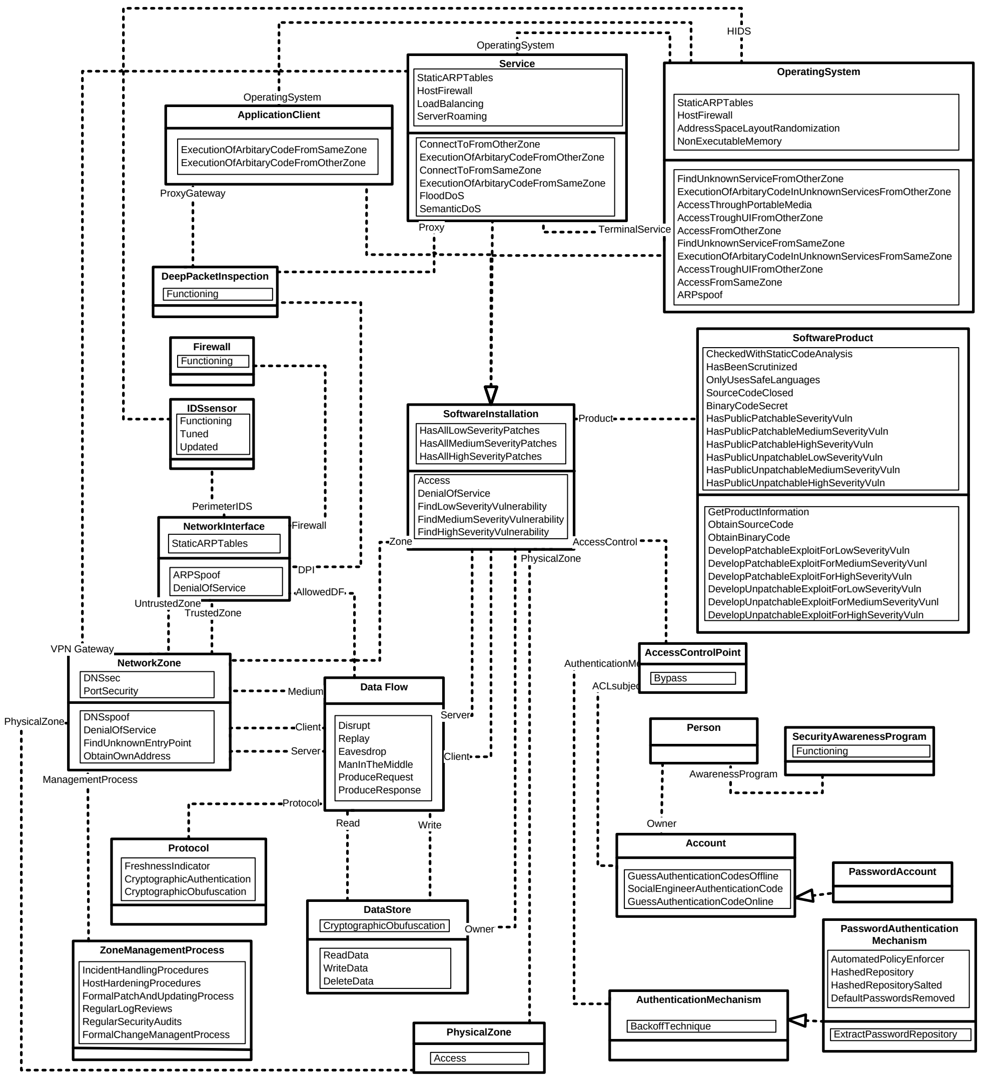


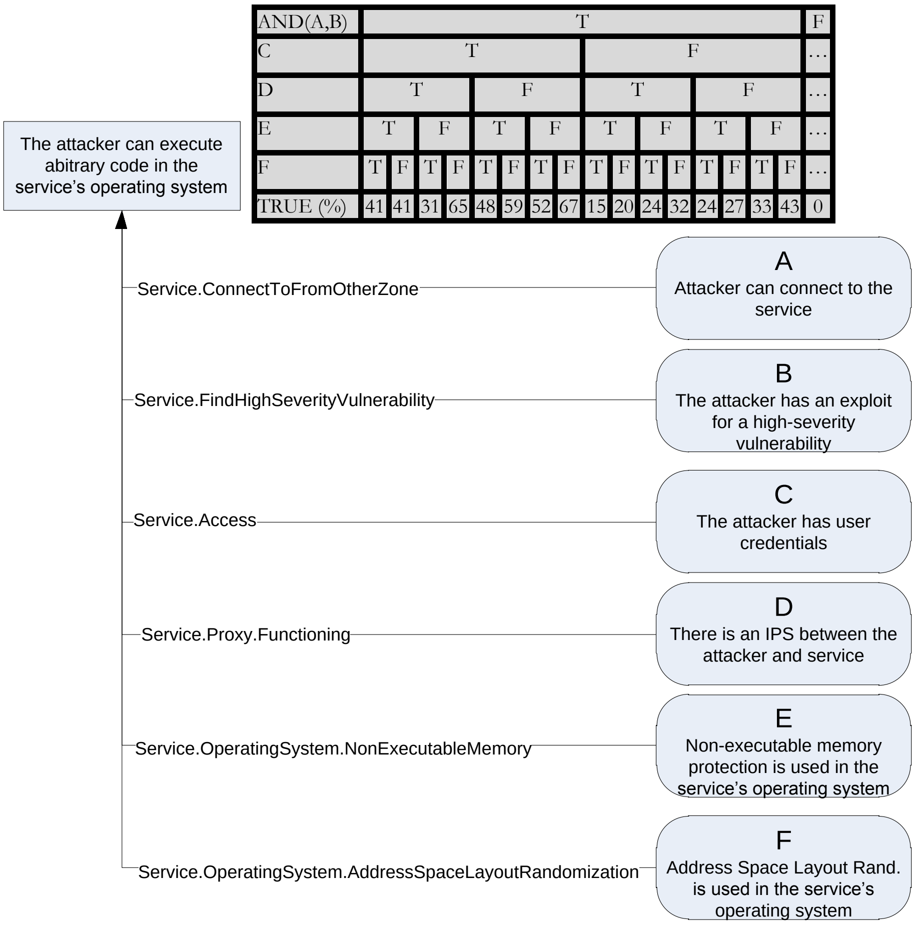
Figure 3. The metamodel of CySeMoL. Countermeasures associated with a class are listed in the class' upper plate. Attack steps associated with a class are listed in the class' lower plate.

3.3 Statements of relationship

CySeMoL describes a large number of relationships. Relationships between classes are expressed as reference slots; relationships between attributes are expressed through slot chains and conditional probability tables. Both types of relationship are directional. The class-relationships (reference slots) are deterministic while many of the attribute-relationships are probabilistic and uncertain.

The attribute-relationships are quantified through conditional probability tables. Just as the constructs are a subset of the constructs in the abstract PRM template, attribute-relationships are a subset of the attribute relationships in the abstract PRM template. This subset is limited to attribute-relationships between subclasses to: *PreventiveCountermeasure* and *AttackStep*, *DetectiveCountermeasure* and *AttackStep*, *ReactiveCountermeasures* and *AttackStep*, *AttackStep* and *AttackStep*. The derived relationships stated in CySeMoL are too many to be described here. Refer to papers B through F for details. An example drawn from paper C is presented in Figure 4. In this example, the influence of six variables is expressed in the conditional probability table. The dependent variable and variables A-C are subclasses to *AttackStep*; variables D-E are subclasses to *PreventiveCountermeasure*. If both parent A and parent B are true, a probabilistic dependency exists. However, if either one of parents A or B is false, the response variable will be false regardless of the state of other variables.

Of all entries in CySeMoL's conditional probability tables, 82 percent are deterministic. In other words, the value is either one or zero under 82 percent of the conditions. Deterministic relationships exist when some set of conditions are required for an attack to be feasible at all (as in the example in Figure 4), or when a variable is used as an aggregate for some other variable to simplify the PRM. The remaining 18 percent of the entries in the conditional probability tables are probabilistic values reflecting uncertainty about the variables state in this scenario. When CySeMoL's theory is applied, it is important to consider this uncertainty. The theory of CySeMoL is specified on a high level of abstraction, and the theory will in many cases only offer a rough approximation.


The attacker can execute abitrary code in the service's operating system
Service.ConnectToFromOtherZone
Service.FindHighSeverityVulnerability
Service.Access
Service.Proxy.Functioning
Service.OperatingSystem.NonExecutableMemory
Service.OperatingSystem.AddressSpaceLayoutRandomization
A
Attacker can connect to the service
B
The attacker has an exploit for a high-severity vulnerability
C
The attacker has user credentials
D
There is an IPS between the attacker and service
E
Non-executable memory protection is used in the service's operating system
F
Address Space Layout Rand. is used in the service's operating system

AND(A,B)	T																F
C	T								F								…
D	T				F				T				F				…
E	T		F		T		F		T		F		T		F		…
F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	…
TRUE (%)	41	41	31	65	48	59	52	67	15	20	24	32	24	27	33	43	0

Figure 4. Examples of relationships stated in CySeMoL.

3.4 Scope

As described in section 3.2, CySeMoL focuses on constructs and relationships that concern the cyber security of SCADA system. This focus influences the relationships that have been included in CySeMoL. However, the relationships that have been included in CySeMoL are equally valid for other domains than SCADA. For instance, the relationships depicted in Figure 4 are general and could be applied to any type of IT system. The studies used to define constructs and relationships have not been limited to the SCADA domain. The theory comes from generic security literature and the judgment of security experts from a broad population. The theory is thus possible to generalize to domains other than SCADA systems.

However, CySeMoL's theory is only valid for a specific threat model. The relationships have been expressed for the case when the threat agent is a professional penetration tester with access to publicly available tools and one week to spend on the attack. Clearly, other threats are also present. For instance, a threat agent can be the unskilled "script kiddie", a well-known computer worm or a group of skilled actors such as a military cyber command. The threat agent may also have access to different toolsets and a different amount of time to spend on the attack. CySeMoL's theory only covers cases concerning the professional penetration tester with publicly available tools and one week to spend.

In addition to delimitations regarding the threat agent the validity of the theory is contingent on developments in the threat environment and the cyber security measures employed in enterprises. Cyber security can be seen as an arms race, where attackers and defenders continuously improve and change their practices [128]. Advances on the attacking side will mean that certain attacks become easier to perform while advances on the defending side will mean that they are more difficult to perform. The theory presented in this thesis marginalizes a considerable number of variables with the assumption that they have the value they typically have in enterprises today. When advances are made on the adversarial side with respect to knowledge, skill, or tools, the estimates will underestimate the capability of attackers on the attack steps in questions. The estimates are also contingent on the assumption that marginalized variables related to enterprises' cyber security practices are as they are today. So, if the average values of architecture-related variables outside the scope of the metamodel change significantly, then the estimates will become less accurate. While this means that the utility of the theory will deteriorate over time, maintaining it should possible if there is a will to do so. For instance, if publicly available tools include techniques to efficiently bypass the operating system protection called address space layout randomization, the validity of relationships where this variable is involved needs to be revised. Similarly, if there is a general increase in the security of software producer's products using means other than those included in this theory, other relationships will need to be revised.

3.5 Causal explanations

The theory in CySeMoL is rich in causal relationships and explanations. All the relationships stated in CySeMoL are drawn from hypotheses concerning causality that are described in the literature. In CySeMoL these are quantified and formally represented. As described in section 3.3, some relationships are probabilistic and some are deterministic. The table in Figure 4 gives examples of both. Textual explanations that further explain the causality are also available. For instance, explanations for the relationships in Figure 4 can be found in paper C. Paper C (like the other papers) also contains references to even more elaborate explanations for why they have a causal influence.

3.6 Testable propositions

An important quality of scientific theory is that it is testable. The propositions concern the capability of a professional penetration tester with one week to spend on this task. This threat is believed to be relevant for decision makers, known well-enough to make theory-construction possible, and possible to test formally to an acceptable extent. However, engaging professional penetration testers in weekly undertakings comes at a cost; formal empirical tests of the propositions put forward in CySeMoL in most cases have a considerable cost associated with them. In fact, the costs and practical obstacles associated with observational studies are the reason why domain experts are used to quantify much of the theory.

Performing experimental tests involving sampled professional penetration testers who spend one week each on an attack is certainly costly. Archival data on attack attempts from the threat agents of the type in question would be an option. However, reliable data of this type is not available today. As a consequence, encompassing tests on all parts of the proposed theory is likely to be costly. However, at a reasonable cost, tests can be performed on selected parts of the theory to test these parts' validity, and tests can be performed on a high level of abstraction on the theory as a whole.

On a low level of abstraction CySeMoL proposes conditional probabilities for specific attack steps (see Figure 4 for an example). A full-fledged experimental setup on this level of abstraction would require a sample of systems where attributes included in CySeMoL correspond to the prediction to be tested, and the attributes not included in CySeMoL are distributed in a way that is representative to those systems used in enterprises today. It also requires a representative sample of penetration testers who are willing to spend a week attacking each system according to a predefined path. Observations can then be made on success-frequencies for all entries in a conditional probability table to assess their calibration. A less resource-demanding approach would be to investigate a few strategically selected table-entries (probabilities) which CySeMoL predicts. Since the conditional probabilities in a table often originate from the same source (e.g., a group of security experts), a test on one entry also indicates the calibration of other entries. Tests arranged with less resourceful threat agents can also falsify the theory. For instance, if less resourceful or less skilled threat agents consistently perform better than CySeMoL predicts this suggests that CySeMoL underestimates the success probability.

On a high level of abstraction, CySeMoL proposes attack paths that have an approximated probability of success. An example is shown in Figure 5. Also on this level of abstraction a full-fledged experimental setup would require representative attackers and sampled system configurations that are representative for an enterprise environment. Like the tests on specific probability values, it also requires a representative sample of penetration testers who are willing to attack each system according to a predefined path. However, tests can be performed on strategically selected attack paths, or with less resourceful and/or competent threat agents. For instance, if threat agents consistently fail attack paths that CySeMoL predicts as easy but succeed with attack paths CySeMoL assigns a marginal success-probability, this would point to validity issues with CySeMoL's theory.

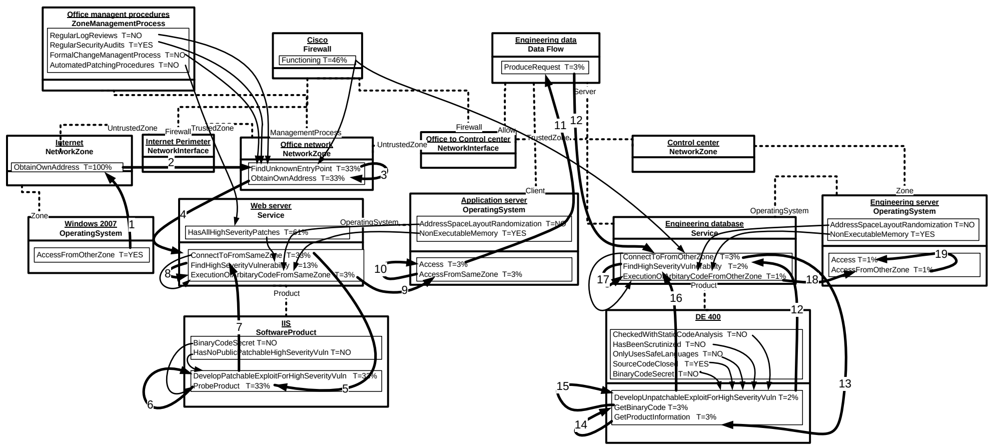


Figure 5. Excerpts from an instance model. A 19-step attack path and probabilities that each step along this path will be reached. The order the path is traversed is shown the enumerated arcs.

Some initial steps have been taken to test and validate the propositions made in CySeMoL through observations. In [129] observations related to remote arbitrary code exploits are made in conjunction with a cyber security exercise, in [96] a formal test of intrusion detection systems' operational effectiveness is made for one scenario and in [130] a formal test is made for one of the propositions CySeMoL makes regarding signature based intrusion detection. These tests corroborates propositions put forward by CySeMoL, however, they only cover a small portion of the theory and only [96] have the threat agent CySeMoL's theory is built around. Yet, they demonstrate the possibility to arrange formal tests of CySeMoL's validity.

A broader test of CySeMoL's convergent validity has been performed by comparing the predictions produced on a high level of abstraction to the predictions made by domain experts concerning a set of system architectures. In the test, the reasonableness of estimates made by CySeMoL was compared to the reasonableness of estimates made by five domain experts and three novices in cyber security. Of the six "experts", CySeMoL ends up in fourth place with respect to mean score, and fifth place with respect to median score. Overall, the test does not show an alarming difference between its ratings and the real experts' ratings. In addition, CySeMoL is rated as more reasonable than all the three novices. This test is further described in paper E.

3.7 Prescriptive statements

The theory of CySeMoL does not prescribe how a decision maker should go about achieving an optimal cyber security solution. The primary reason for this is that the theory does not include a number of variables that are required when the utility of a solution is to be assessed, including:

a) The consequence of attacks and the influence of contingency measures on this consequence, for instance, the cost of an unavailable SCADA server.
b) All threat agents that are relevant for a decision maker, for instance, insiders within SCADA system suppliers or undirected malicious code.
c) The mental model of threat agents and how often they attempt attacks of different types, for instance, how often they are likely to attempt attacks involving social engineering.
d) The business value (or cost) associated with different architectures, for instance, the value of making historical measurements available to IT systems in administrative office networks.

The abstract PRM template suggests how theories on a), b), and c) could be integrated with the theory presented in this thesis. The output of a theory that encompasses all constructs in the abstract PRM template could then be contrasted to the output of methods that assess the business value of an enterprise architecture, i.e., paragraph d). For instance, the method described in [131] could be used.

While important variables are outside the scope of the theory, and CySeMoL cannot be used to produce prescriptive statements directly, the theory can be used to produce prescriptive statements when these variables values have been assessed. The vulnerability estimates produced by CySeMoL can also be used to produce prescriptive statements ceteris paribus. Clearly, a less vulnerable architecture is desirable if all other variables remain unchanged. When perceptive statements are produced it is important to remember that CySeMoL produces rough approximations. It does not produce exact success probabilities.

4 Research design

This section gives an overview of the methodological aspects that have guided the research. The description is process-oriented and each sub-section corresponds to a phase in the research. These phases are (cf. Figure 6): framework and formalism, qualitative theory, quantitative theory and validation. The methods used for data collection and analysis within each of these phases are described.

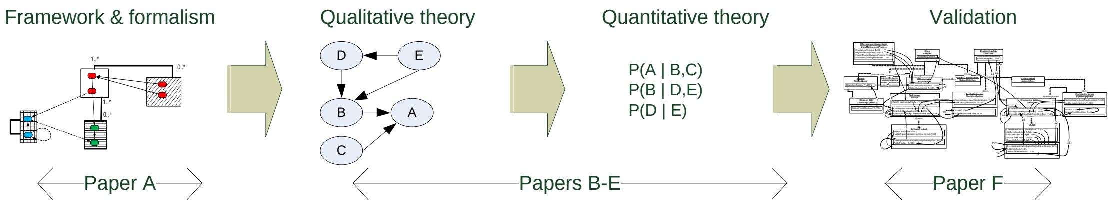


Figure 6. Phases in the research.

4.1 Framework and formalism

The primary purpose of this research is to support decision makers when they need to assess the cyber security of their SCADA systems. While the cyber security issues pertaining to SCADA systems are fairly new, a substantial theoretical body is available with the security field as a whole. This research reviewed existing literature in the field and compared it with the needs of decision makers in the SCADA domain. A number of methods and models have been proposed to address the problem of measuring cyber security, however, none of these were found to fit the needs in their present state (section 2 explained why).

Literature was the primary information source used when the framework used in this research was developed. The result combined qualitative models found in literature with a mathematical formalism and puts these into a framework which allows causal cyber security theory to be coupled with architectural models. As this framework was used as a basis, it has an influence on the approach used in other parts of this research. The framework approaches cyber security assessments as risk assessments and aims at quantifying the monetary risk associated with different architectures, i.e., the probability of unwanted events and the expected consequences of these events. The framework also directs the theory developer to model the attacks that give rise to the risk and the influence of countermeasures that reduce it. The primary sources of inspiration for this framework are Common Criteria's and its conceptual model [21], time-based-security [27], attack-modeling [32], [113], [114] and monetary security risk assessments [40], [132]. The formalism used to couple this framework to architectural models was that of PRMs [123]. The result was the abstract PRM template described in section 3.1 and paper A.

4.2 Qualitative theory

The framework (or PRM template) was used to develop a qualitative theory over cyber security. This qualitative theory details the PRM's: classes, reference slots, attributes and attribute relationships. In other words, it details everything except the conditional probabilities of the PRM.

An extensive literature review and interviews with experts in the cyber security domain were the primary sources for this theory. The objective was to produce a qualitative causal theory to support assessments of cyber security vulnerability. A subset of the framework was used for this purpose. To efficiently tackle practical issues relating to cyber security assessments this theory should offer a good tradeoff between the cost of applying the theory, the cost of quantifying the theory and the theory's accuracy.

First, literature was consulted to identify which attack steps to include. This literature study included review of a large number of textbooks (e.g. [133]), standards and reports (e.g. [9]), overview-articles (e.g. [104]) and security databases (e.g. [134]). After an initial model over attacks and assets had been created, literature on specific attacks was consulted. These sources were used to assess the parents to attack steps, i.e., countermeasures and states (completed attack steps) that literature suggests have an important influence on the probability that an attack step could be accomplished. A large number of sources were used for each type of attack. Examples of sources can be found in section 2.2 and in papers B-F.

The qualitative model was subsequently reviewed by domain experts. These reviews were made both on a high level of abstraction to ensure that the scope constituted a reasonable tradeoff and on a low level of abstraction to prioritize specific countermeasures and operationalize their definitions. Overall, these experts confirmed the prioritizations that had been made based on literature, but suggested some minor changes, e.g., to focus more on attacks on password authentication. For the reviews on a low level of abstraction, the number of reviewers used varied with the attack type. For instance, literature on social engineering was deemed sufficient to prioritize this field, while the details on remote code exploits was decided after a pilot study was made and after consulting three domain experts. Details concerning the expert reviews can be found in papers B-F.

4.3 Quantitative theory

The qualitative theory describes the relationships that need to be quantified. A large portion of the relationships could be quantified from the definition of constructs. An example of such a definitional relationship is that an attacker must possess an exploit code if he/she is to exploit a software vulnerability in a remote service. The relationships that cannot be determined from the definition of constructs were analyzed as in "probabilistic causal analysis" [122]. In other words, it was perceived as difficult to identify and control all variables that may influence the response variable's state. Since relevant variables are missing from the analysis the causal effect becomes uncertain (and probabilistic). In Bayesian terms, the omitted variables can be seen as marginalized [124].

Two methods were employed to assess probabilities. When reliable data could be found in the literature this data was used. When no reliable approximations could be found, data was elicited from domain experts.

Searches for data in literature were performed in article indexing services (e.g., Scopus and Google Scholar). They aimed at finding studies that contained data on the relationships specified in the qualitative theory. To quantify a relationship using secondary data the study should not only be of sufficient quality, but the variables studies should also match the variables and variable-relationships prescribed in the qualitative model. A number of relationships were possible to quantify using quantitative data from previous research in the field. Research on password security ([135–138]), network misconfigurations ([56], [57]) and social engineering ([52–55]) was directly used to determine variables' probability distributions given the conditions specified.

When the literature review was unable to find the data required it appeared not to be because the research community had ignored the relationship in question. The problem was rather that is was difficult for a researcher to quantify the relationship through observation in a manner that made the result generalizable. For instance, testing intrusion detection systems is associated with a number of issues, such as producing representative attacks and representative background traffic [88], [90]. In order to produce a quantitative theory that could approximate these relationships the judgment of domain experts was used.

Experts in the scientific community were the primary respondents in these surveys. However, a number of practitioners were also included. Researchers were identified from their publications; practitioners were identified based on peer-recommendation. Web surveys were used as the elicitation instrument. Since estimation of probability distributions is known to be problematic [139] care was taken with the construction of the web survey. The reliability of the question format was confirmed using Cronbach's alpha [140], [141] and all surveys were qualitatively reviewed by members of the target population.

Research in the field of expert judgment elicitation suggests that the result is better calibrated when multiple experts are used [142]. A number of techniques has been suggested for combining expert judgment, including: equal-weight, consensus methods [143], [144], the Cochran-Weiss-Shanteau index [145], self-proclaimed expertise [146], experience [147], certifications [147], peer-recommendations [147], and Cooke's classical method [148]. There is little research that compares the accuracy that these methods yield. This research uses the scheme proposed in Cooke's classical method [148]. Cooke's classical method has been shown to outperform both the best expert in a group, and the equal-weight combination of all experts' assessments. It is a performance based method which assigns weights based on the experts' ability to answer a set of test questions (called "seed questions") in a calibrated (i.e., accurate) and informative (i.e., precise) way. In the presented research these questions were constructed from previous research results in the field in question.

More elaborate descriptions of the elicitation process and the implementation of Cooke's classical method are given in papers B-E.

4.4 Validation

The interviews undertaken during theory development provided a qualitative validation of the relationships included in the theory. The surveys described in papers B-E also validated the prioritizations underlying the theory by asking respondents to suggest improvements. The few changes suggested by the respondents were diverse. In addition to this validation, CySeMoL's practical utility has been validated in three case studies, and the reasonableness of its assessments has been validated with a variant of the Turing test.

The scopes of the three case studies were: (1) the control center and adjacent environments in one of Sweden's three largest electrical power utilities, (2) electrical substations and remote communication to these owned by one of Sweden's largest power system owners and (3) reference architectures for one of the world's most commonly used electrical power management systems. The case studies demonstrated that the theory served as a usable tool for architecture analysis and pointed to practical improvements which would increase usability of the software tool.

A variant of the Turing test was used to test CySeMoL's validity [149]. In the classical Turing test a machine shall behave in a way indistinguishable from humans. These tests are especially useful for testing expert systems in situations such as the present – where the true answers to test cases are unknown (or very costly to determine), and it cannot be assumed that one particular domain expert is correct [150]. The test of CySeMoL was similar to the tests described in [68] and [71] and had two pools of human experts: one that produced assessments of the same type as the expert system and one that evaluated the first pool's assessments and the expert system's assessments based on how reasonable they are. The idea is that the expert system (i.e., CySeMoL) should receive ratings for the evaluators that are similar to the ratings received by the real experts. To test if the evaluators could recognize expertise, the test also included a pool of information system experts which were novices in the cyber security field. These novices' assessments were evaluated in the same manner as the assessments made by the experts and CySeMoL. If the evaluators recognize expertise the novices should receive comparably low ratings.

The pool of experts that produced assessments of the same type as CySeMoL consisted of five persons. The pool of cyber security novices consisted of three persons, and the pool that rated the assessments reasonableness consisted of two persons. The sample size prohibits reliable statistical conclusions from this test. The variation between the evaluators' scoring of the solutions suggests that the result should be interpreted with care. However, the summary statistics indicates that CySeMoL's assessments are comparable to those of a domain expert. In terms of mean score CySeMoL's comes in a tied fourth place; in terms of median score CySeMoL is placed on fifth. It also appears as if the evaluators' ratings are meaningful – there is a clear difference between the ratings that novices receive and the ratings that experts receive.

A more thorough description of the qualitative validation made on variables and relationships can be found in papers A-E. In paper F a more thorough description of the validation Turing test is given.

As described in 3.6, some initial attempts were made to validate the theory through formal experiments. In [96], [130] two experiments concerning intrusion detection systems are described. In [96] a formal test of intrusion detection systems' operational effectiveness is made. This test roughly corresponds to one of the intrusion detection scenarios in CySeMoL. The test in [96] gave a detection rate of 58 percent, and the value CySeMoL predicts is 59 percent. In [130] a formal test is made concerning the possibility to detect zero-day attacks (i.e., new and novel attacks) with signature based intrusion detection systems. As predicted by CySeMoL (c.f. paper D) it shows that signature based systems can detect zero-day attacks. In addition to these experiments [129] describes less reliable observations made in conjunction to a cyber security exercise. The observations concern remote arbitrary code exploits performed by a different threat agent under tighter time-constraints than about which the threat agent CySeMoL makes predictions. The observations made in [129] correspond to two scenarios predicted in CySeMoL's theory (one variable in CySeMoL is unknown for the observations). CySeMoL predicts these two scenarios to be successful with 43 percent and 67 percent probability while the observed frequency was 33 percent. Since the observed threat agent was less resourceful than the one CySeMoL makes predictions about the lower value offers some (albeit weak) support for CySeMoL's theory. Additional testing and refinement of CySeMoL's theory is suggested as future work.

5 References

[1] T. Cegrell, *Power system control technology*. Cambridge, Great britain: Prentice hall International, 1986.

[2] J. Andersson, E. Johansson, M. Haglind, and L. Johansson, "State-of-the-art study of commercial industrial control systems," Royal Institute of Technology - KTH, Stockholm, Sweden, 1997.

[3] GLEG Ltd, "Agora Exploit Pack Developer- SCADA+ pack," *(website)*, 2012. [Online]. Available: http://www.gleg.net/agora_scada.shtml. [Accessed: 20-Jun-2012].

[4] T. Sommestad, G. Björkman, and M. Ekstedt, "Information System Architectures in Electrical Distribution Utilities," in *Proceedings of NORDAC 2010*, 2010.

[5] E. Johansson, T. Sommestad, and M. Ekstedt, "Issues of Cyber Security In Scada-Systems-on the Importance of Awareness," in *The 20th International Conference on Electricity Distribution (CIRED)*, 2009.

[6] R. Fink, D. Spencer, and R. Wells, "Lessons learned from cyber security assessments of SCADA and energy management systems," *US Department of Energy*, no. September, 2006.

[7] U.S. Department of Energy, "Common Cyber Security Vulnerabilities Observed in Control System Assessments by the INL NSTB Program," Idaho Falls, 2008.

[8] US Department of Homeland Security, "Catalog of Control Systems Security: Recommendations for Standards Developers," 2008.

[9] K. Stouffer, J. Falco, and K. Kent, "Guide to Industrial Control Systems (ICS) Security Recommendations of the National Institute of Standards and Technology," *Nist Special Publication 800- 82*, 2008.

[10] IEC, "TS 62351-1: Power systems management and associated information exchange Data and communications security, Part 1:Communication network and system security Introduction to security issues," Geneva,Switzerland, 2007.

[11] NERC, "NERC CIP 002-009," 2007.

[12] TOGAF, "The Open Group Architecture Framework (TOGAF) - version 9." The Open Group, 2009.

[13] MODAF, "MOD Architecture Framework (MODAF)," 2012. [Online]. Available: www.modaf.org.uk. [Accessed: 17-Jul-2012].

[14] Department of Defense Architecture Framework Working Group, "DoD Architecture Framework, version 1.5," 2007.

[15] P. Johnson, L. Nordström, and R. Lagerström, "Formalizing analysis of enterprise architecture," in *Interoperability for Enterprise Software and Applications Conference*, 2006, p. 10.

[16] P. Johnson, R. Lagerström, P. Närman, and M. Simonsson, "Extended Influence Diagrams for System Quality Analysis," *Journal of Software*, vol. 2, no. 3, pp. 30-42, Sep. 2007.

[17] MODAF, "FAQs: How does MODAF represent security?," 2008. [Online]. Available: http://www.mod.uk/NR/rdonlyres/6F2454B0-48A3-4E61-90A4-F420CF9F3F1C/0/20090521_MODAF_1_2_FAQs_How_MODAF_Can_Reflect_Security_Concerns_V1_0_U.pdf. [Accessed: 17-Jul-2012].

[18] International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC)., "ISO/IEC 27004: Information technology -- Security techniques -- Information security management -- Measurement.," 2009.

[19] M. Swanson, N. Bartol, J. Sabato, J. Hash, and Laurie Graffo, "Security Metrics Guide for Information Technology Systems," *NIST Special Publications*, vol. 800, no. 55, 2003.

[20] M. S. Lund, B. Solhaug, and K. Stolen, *Model-driven risk analysis: the CORAS approach*. Springer Verlag, 2011.

[21] "Common Criteria for Information Technology Security Evaluation Part 2 : Security functional components September 2007 Revision 2 Foreword," no. September, pp. 1-324, 2007.

[22] E. Johansson, "Assessment of Enterprise Information Security–How to make it Credible and efficient," KTH - The Royal Insitute of Technology, 2005.

[23] J. Sherwood, A. Clark, and D. Lynas, *Enterprise Security Architecture: A Business-Driven Approach*. USA: CMP Books, 2005.

[24] G. Stoneburner, "Underlying Technical Models for Information Technology Security," *Nist Special Publications 800-33*, Dec. 2001.

[25] A. Anderson, D. Longley, and L. F. Kwok, "Security modelling for organisations," in *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, 1994, pp. 241-250.

[26] D. Bodeaum, "A conceptual model for computer security risk analysis," in *Proceedings Computer Security Applications Conference, 1992. ., Eighth Annual*, 1992, pp. 56–63.

[27] W. Schwartau, "Time-based security explained: Provable security models and formulas for the practitioner and vendor," *Computers & Security*, vol. 17, no. 8, pp. 693-714, 1998.

[28] L. Pirzadeh and E. Jonsson, "A Cause and Effect Approach towards Risk Analysis," *2011 Third International Workshop on Security Measurements and Metrics*, pp. 80-83, Sep. 2011.

[29] E. Jonsson, "Towards an integrated conceptual model of security and dependability," *First International Conference on Availability, Reliability and Security*, pp. 646-653, 2006.

[30] M. Beccuti et al., "Quantification of dependencies in electrical and information infrastructures: The CRUTIAL approach," in *2009 Fourth International Conference on Critical Infrastructures*, 2009, pp. 1-8.

[31] T. Fleury, H. Khurana, and V. Welch, "Towards a taxonomy of attacks against energy control systems," in *Critical Infrastructure Protection II*, no. March 2008, Springer US, 2009, p. 71-85.

[32] B. Schneier, "Attack trees: Modeling security threats," *Dr. Dobb's Journal*, 1999.

[33] M. Howard and D. C. LeBlanc, *Writing Secure Code*. Redmond, WA, USA: Microsoft Press, 2002.

[34] S. Bistarelli, F. Fioravanti., and P. Peretti., "Defense trees for economic evaluation of security investments," in *Proceedings of the First International Conference on Availability, Reliability and Security* , pp. 416-423, 2006.

[35] K. Edge, R. Raines, M. Grimaila, R. Baldwin, R. Bennington, and C. Reuter, "The Use of Attack and Protection Trees to Analyze Security for an Online Banking System," *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, p. 144b-144b, Jan. 2007.

[36] L. Piètre-Cambacédès and M. Bouissou, "Beyond Attack Trees: Dynamic Security Modeling with Boolean Logic Driven Markov Processes (BDMP)," *2010 European Dependable Computing Conference*, pp. 199-208, 2010.

[37] J. Hallberg, N. Hallberg, and A. Hunstad, "Crossroads and XMASS: Framework and method for system it security assessment," Linköping, Sweden, 2006.

[38] B. Karabacak and I. Sogukpinar, "ISRAM: information security risk analysis method," *Computers & Security*, vol. 24, no. 2, pp. 147-159, 2005.

[39] G. Stoneburner, A. Goguen, and A. Feringa, "Risk management guide for information technology systems," *NIST Special Publication 800-3*0, 2002.

[40] L. A. Gordon and M. P. Loeb, *Managing Cybersecurity Resources: A Cost-Benefit Analysis*, vol. The Mcgraw. New York, NY, USA: Mcgraw-Hill, 2006.

[41] H. Mouratidis, P. Giorgini, G. Manson, and I. Philp, "A natural extension of tropos methodology for modelling security," in *the Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002)*, 2002.

[42] C.-W. Ten, C.-C. Liu, and M. Govindarasu, "Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees," *2007 IEEE Power Engineering Society General Meeting*, vol. 2, pp. 1-8, Jun. 2007.

[43] M. McQueen, W. Boyer, S. McBride, M. Farrar, and Z. Tudor, "Measurable Control System Security through Ideal Driven Technical Metrics," in *S4: SCADA Security Scientific Symposium*, 2008.

[44] G. Dondossola, F. Garrone, and J. Szanto, "Cyber Risk Assessment of Power Control Systems – A Metrics weighed by Attack Experiments," in *Proceedings of IEEE Power and Energy Society General Meeting*, pp. 1-9, 2011.

[45] R. Breu, F. Innerhofer-Oberperfler, and A. Yautsiukhin, "Quantitative Assessment of Enterprise Security System," *2008 Third International Conference on Availability, Reliability and Security*, pp. 921-928, Mar. 2008.

[46] E. J. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," in *International Infrastructure Survivability Workshop (IISW'04)*, 2004.

[47] I. Nai Fovinoa, A. Carcanoa, M. Maseraa, and A. Trombettab, "An experimental investigation of malware attacks on SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 4, pp. 139-145, 2009.

[48] G. Dondossola, J. Szanto, M. Masera, and I. N. Fovino, "Effects of intentional threats to power substation control systems," *International Journal of Critical Infrastructures*, vol. 4, no. 1/2, p. 129, 2008.

[49] N. SCADA and U.S. Department of Energy, "Common Cyber Security Vulnerabilities Observed in Control System Assessments by the INL NSTB Program," Idaho Falls, 2008.

[50] G. Dondossola, F. Garrone, and J. Szanto, “Assessment of power control systems communications through testbed experiments,” in *Electricity Distribution - Part 1, 2009. CIRED 2009. 20th International Conference and Exhibition on*, 2009, no. 0650, pp. 1-4.

[51] W. Jansen, “Directions in security metrics research,” DIANE Publishing, Gaithersburg, MD, 2009.

[52] J. R. Jacobs, “Measuring the Effectiveness of the USB Flash Drive as a Vector for Social Engineering Attacks on Commercial and Residential Computer Systems,” Embry Riddle Aeronautical University, 2011.

[53] S. Stasiukonis, “Social engineering, the USB way,” *Dark Reading*, vol. 7, 2006.

[54] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, “Social phishing,” *Communications of the ACM*, vol. 50, no. 10, pp. 94–100, Mar. 2007.

[55] R. Dodge and A. Ferguson, “Using Phishing for User Email Security Awareness,” in *Security and Privacy in Dynamic Environments*, vol. 201, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds. Springer Boston, 2006, pp. 454-459.

[56] T. Sommestad, M. Ekstedt, H. Holm, and M. Afzal, “Security mistakes in information system deployment projects,” *Information Management and Computer Security*, vol. 19, no. 2, 2011.

[57] A. Wool, “A quantitative study of firewall configuration errors,” *Computer*, pp. 62–67, 2004.

[58] NIST Computer Security Resource Center (CSRC), “National Vulnerability Database,” 2011. [Online]. Available: www.nvd.nist.org. [Accessed: 28-Apr-2011].

[59] The MITRE Corporation, “Common Weakness Enumeration,” 2012. [Online]. Available: http://cwe.mitre.org/.

[60] Offensive Security, “Exploit Database,” 2011. [Online]. Available: http://www.exploit-db.com/.

[61] A. Ozment, “Improving vulnerability discovery models,” in *Proceedings of the 2007 ACM workshop on Quality of protection*, 2007, pp. 6–11.

[62] S.-W. Woo, H. Joh, O. H. Alhazmi, and Y. K. Malaiya, “Modeling vulnerability discovery process in Apache and IIS HTTP servers,” *Computers & Security*, vol. 30, no. 1, pp. 50-62, Jan. 2011.

[63] O. H. Alhazmi and Y. K. Malaiya, “Modeling the Vulnerability Discovery Process,” *16th IEEE International Symposium on Software Reliability Engineering (ISSRE’05)*, pp. 129-138, 2005.

[64] O. H. Alhazmi and Y. K. Malaiya, “Quantitative vulnerability assessment of systems software,” in *Proceedings of Annual Reliability and Maintainability Symposium*, 2005, pp. 615-620.

[65] T. Gerace and H. Cavusoglu, "The critical elements of the patch management process," *Communications of the ACM*, vol. 52, no. 8, p. 117, Aug. 2009.

[66] B. David, P. Pongsin, S. Dawn, and Z. Jiang, "Automatic patch-based exploit generation is possible: Techniques and implications," *IEEE Symposium on Security and Privacy*, pp. 143-157, May 2008.

[67] M. A. McQueen, T. A. McQueen, W. F. Boyer, and M. R. Chaffin, "Empirical estimates and observations of 0day vulnerabilities," in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, 2009, pp. 1–12.

[68] J. Wilander and M. Kamkar, "A comparison of publicly available tools for dynamic buffer overflow prevention," in *Proceedings of the 10th Network and Distributed System Security Symposium*, 2003, pp. 149–162.

[69] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole, "Buffer Overflows : Attacks and Defenses for the Vulnerability of the Decade," in *Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems]*, 2003, pp. 227-237.

[70] I. Simon, "A comparative analysis of methods of defense against buffer overflow attacks," *Web address: http://www. mcs. csuhayward. edu/\~ simon/security/boflo. html*, pp. 1-16, 2001.

[71] N. Frykholm, "Countermeasures against buffer overflow attacks," *RSA Tech Note*, pp. 1-9, 2000.

[72] Y. Kim, J. Lee, H. Han, and K.-M. Choe, "Filtering false alarms of buffer overflow analysis using SMT solvers," *Information and Software Technology*, vol. 52, no. 2, pp. 210-219, Feb. 2010.

[73] X. Wang, C. C. Pan, P. Liu, and S. Zhu, "SigFree: A Signature-Free Buffer Overflow Attack Blocker," | *IEEE Transactions on Dependable and Secure Computing*, pp. 65–79, 2008.

[74] B. Salamat, A. Gal, T. Jackson, K. Manivannan, G. Wagner, and M. Franz, "Multi-variant Program Execution: Using Multi-core Systems to Defuse Buffer-Overflow Vulnerabilities," in *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, 2008, pp. 843-848.

[75] S. H. Yong and S. Horwitz, "Protecting C programs from attacks via invalid pointer dereferences," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 5, p. 307, Sep. 2003.

[76] J. Wilander, N. Nikiforakis, Y. Younan, M. Kamkar, and W. Joosen, "RIPE: Runtime Intrusion Prevention Evaluator," in *In Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC*, 2011.

[77] Y. Younan, "Efficient countermeasures for software vulnerabilities due to memory management errors," Katholieke Universiteit Leuven, 2008.

[78] H. Shacham, M. Page, B. Pfaff, E. J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space

randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298–307.

[79] PaX. Team, "PaX address space layout randomization (ASLR)." [Online] 2003. http://pax.grsecurity.net/docs/aslr.txt

[80] U. Erlingsson, "Low-level Software Security : Attacks and Defenses Low-level Software Security : Attacks and Defenses," Redmond, WA, USA, 2007.

[81] T. Jim, G. Morrisett, D. Grossman, M. Hicks, J. Cheney, and Y. Wang, "Cyclone: A safe dialect of C," in *USENIX*, 2002, pp. 275-288.

[82] J. Newsome, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," *Network and Distributed System Security*, no. May 2004, 2005.

[83] R. Lippmann et al., "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, pp. 12-26, 1998.

[84] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA on-line intrusion detection evaluation," *Computer Networks*, vol. 34, 2000.

[85] K. Salah and a. Kahtani, "Improving Snort performance under Linux," *IET Communications*, vol. 3, no. 12, p. 1883, 2009.

[86] F. Alserhani, M. Akhlaq, I. U. Awan, J. Mellor, A. J. Cullen, and P. Mirchandani, "Evaluating Intrusion Detection Systems in High Speed Networks," *2009 Fifth International Conference on Information Assurance and Security*, pp. 454-459, 2009.

[87] F. B. Ktata, N. E. Kadhi, and K. Ghédira, "Agent IDS based on Misuse Approach," *Journal of Software*, vol. 4, no. 6, pp. 495-507, Aug. 2009.

[88] J. McHugh, "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262-294, Nov. 2000.

[89] B. I. A. Barry and H. A. Chan, "Intrusion detection systems," in *Handbook of Information and Communication Security*, vol. 2001, no. 6, P. Stavroulakis and M. Stamp, Eds. Springer, 2010, pp. 193-205.

[90] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, "An overview of issues in testing intrusion detection systems," *Citeseer*. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2003.

[91] M. J. Ranum, "Experiences Benchmarking Intrusion Detection Systems," *Security*, pp. 1-10, 2001.

[92] R. Werlinger, K. Hawkey, and K. Muldner, "The challenges of using an intrusion detection system: is it worth the effort?," *SOUPS '08 Proceedings of the 4th symposium on Usable privacy and security*, no. 1, 2008.

[93] R. Werlinger, K. Muldner, K. Hawkey, and K. Beznosov, "Preparation, detection, and analysis: the diagnostic work of IT security incident response," *Information Management & Computer Security*, vol. 18, no. 1, pp. 26–42, 2010.

[94] J. R. Goodall, W. G. Lutters, and A. Komlodi, "I know my network: collaboration and expertise in intrusion detection," in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004, pp. 342–345.

[95] J. R. Goodall, W. G. Lutters, and A. Komlodi, "Developing expertise for network intrusion detection," *Information Technology & People*, vol. 22, no. 2, pp. 92–108, 2009.

[96] T. Sommestad and A. Hunstad, "Intrusion detection and the role of the system administrator," in *Proceedings of International Symposium on Human Aspects of Information Security & Assurance*, 2012.

[97] R. Chertov, S. Fahmy, and N. B. Shroff, "Emulation versus simulation: A case study of TCP-targeted denial of service attacks," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, 2006, p. 10–pp.

[98] C. Sangpachatanaruk, S. M. Khattab, T. Znati, R. Melhem, and D. Mossé, "A simulation study of the proactive server roaming for mitigating denial of service attacks," in *Proceedings of the 36th annual symposium on Simulation*, 2003, p. 7.

[99] S. M. Khattab, C. Sangpachatanaruk, R. Melhem, and T. Znati, "Proactive server roaming for mitigating denial-of-service attacks," in *Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on*, 2003, pp. 286–290.

[100] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115-139, May 2006.

[101] V. Gupta, S. Krishnamurthy, and M. Faloutsos, "Denial of service attacks at the MAC layer in wireless ad hoc networks," in *MILCOM 2002*, 2002, vol. 2, pp. 1118–1123.

[102] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites," in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 293–304.

[103] J. Mirkovic et al., "Testing a Collaborative DDoS Defense In a Red Team/Blue Team Exercise," *IEEE Transactions on Computers*, vol. 57, no. 8, pp. 1098-1112, Aug. 2008.

[104] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, p. 39, Apr. 2004.

[105] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[106] C. Douligeris, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643-666, Apr. 2004.

[107] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computing Surveys*, vol. 39, no. 1, p. 3-es, 2007.

[108] M. Glenn, "A Summary of DoS/DDoS Prevention, Monitoring and Mitigation Techniques in a Service Provider Environment," 2003.

[109] K. A. Lee, "CS2SAT : The Control Systems Cyber Security Self-Assessment Tool Tool," Idaho Falls, Idaho, USA, 2008.

[110] US-CERT, "Cyber Security Evaluation Tool (CSET)," 2012. [Online]. Available: http://www.us-cert.gov/control_systems/satool.html. [Accessed: 30-Apr-2012].

[111] T. Heberlein, M. Bishop, E. Ceesay, M. Danforth, and CG, "A Taxonomy for Comparing Attack-Graph Approaches," *netsq.com*, pp. 1-14, 2004.

[112] S. Roschke, F. Cheng, R. Schuppenies, and C. Meinel, "Towards Unifying Vulnerability Information for Attack Graph Construction," in *Proceedings of the 12th International Conference on Information Security*, 2009, p. 233.

[113] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, 2000, pp. 307-321.

[114] O. M. Sheyner, "Scenario graphs and attack graphs," Carnegie Mellon University, 2004.

[115] R. Lippmann, "Netspa: A network security planning architecture," Massachusetts Institute of Technology, 2002.

[116] R. Lippmann et al., "Validating and restoring defense in depth using attack graphs," in *MILCOM*, pp. 1-10, 2006.

[117] S. Jajodia, "Topological analysis of network attack vulnerability," *Proceedings of the 2nd ACM symposium on Information, computer and communications security - ASIACCS '07*, p. 2, 2007.

[118] S. Jajodia, S. Noel, and B. O'Berry, "Topological analysis of network attack vulnerability," in *Managing Cyber Threats: Issues, Approaches and Challanges, chapter 5. Kluwer Academic Publisher*, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. Springer US, 2003, pp. 247-266.

[119] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, and K. Prole, *Advances in Topological Vulnerability Analysis*. Washington, DC: IEEE, 2009, pp. 124-129.

[120] J. Homer, K. Manhattan, X. Ou, and D. Schmidt, "A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks," Kansas State University, 2010.

[121] H. Holm, T. Sommestad, J. Almroth, and M. Persson, "A quantitative evaluation of vulnerability scanning," *Information Management & Computer Security*, vol. 19, no. 4, 2011.

[122] S. Gregor, "The nature of theory in information systems," *Management Information Systems Quarterly*, pp. 1-45, 2006.

[123] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar, "Probabilistic Relational Models," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2007, pp. 129-175.

[124] F. . Jensen, *Bayesian Networks and Decision Graphs*. Secaucus, NJ, USA.: Springer New York, 2001.

[125] OMG, "OMG Unified Modeling Language (OMG UML), Infrastructure," 2009.

[126] F. . Jensen, *Bayesian Networks and Decision Graphs*. Secaucus, NJ, USA.: Springer New York, 2001.

[127] P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System (CVSS), Version 2.0, Forum of Incident Response and Security Teams." 2007.

[128] D. Ahmad, "The Contemporary Software Security Landscape," *IEEE Security & Privacy Magazine*, vol. 5, no. 3, pp. 75-77, May 2007.

[129] H. Holm, T. Sommestad, U. Franke, and M. Ekstedt, "Success rate of remote code execution attacks – expert assessments and observations," *Journal of Universal Computer Science*, vol. 18, no. 6, pp. 732-749, 2012.

[130] H. Holm, "Empirical analysis of signature based intrusion detection for zero-day exploits," *Working paper,* Royal Institute of Technology, 2012.

[131] M. Gammelgård, "Business value assessment of it investments an evaluation method applied to the electrical power industry," Royal Institute of Technology (KTH), 2007.

[132] H. Cavusoglu, B. Mishra, and S. Raghunathan, "A model for evaluating it security investments," *Communications of the ACM*, vol. 47, no. 7, pp. 87-92, 2004.

[133] R. J. Anderson, *Security Engineering: A guide to building dependable distributed systems*. New York, NY, USA: Wiley Publishing, 2008.

[134] The MITRE Corporation, "The Common Attack Pattern Enumeration and Classification," *(website)*, 2011. [Online]. Available: http://capec.mitre.org/.

[135] S. Marechal, "Advances in password cracking," *Journal in Computer Virology*, vol. 4, no. 1, pp. 73-81, 2007.

[136] M. Dell' Amico, P. Michiardi, and Y. Roudier, "Password Strength: An Empirical Analysis," *2010 Proceedings IEEE INFOCOM*, pp. 1-9, Mar. 2010.

[137] J. Cazier, "Password security: An empirical investigation into e-commerce passwords and their crack times," *Information Security Journal: A Global*, 2006.

[138] "Free Rainbow Tables," 2011. [Online]. Available: http://www.freerainbowtables.com/. [Accessed: 01-Apr-2011].

[139] P. H. Garthwaite, J. B. Kadane, and A. O'Hagan, "Statistical methods for eliciting probability distributions," *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 680-701, 2005.

[140] L. J. Cronbach and R. J. Shavelson, "My Current Thoughts on Coefficient Alpha and Successor Procedures," *Educational and Psychological Measurement*, vol. 64, no. 3, pp. 391-418, Jun. 2004.

[141] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297-334, 1951.

[142] R. T. Clemen and R. L. Winkler, "Combining probability distributions from experts in risk analysis," *Risk Analysis*, vol. 19, no. 187, pp. 187-204, 1999.

[143] A. Fink, J. Kosecoff, M. Chassin, and R. H. Brook, "Consensus methods: characteristics and guidelines for use.," *American journal of public health*, vol. 74, no. 9, pp. 979-83, Sep. 1984.

[144] A. H. Ashton, "Does consensus imply accuracy in accounting studies of decision making?," *The Accounting Review*, vol. 60, no. 2, pp. 173–185, 1985.

[145] D. J. Weiss and J. Shanteau, "Empirical Assessment of Expertise," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 1, pp. 104-116, 2003.

[146] M. J. Abdolmohammadi and J. Shanteau, "Personal attributes of expert auditors," *Organizational Behavior and Human Decision Processes*, vol. 53, no. 2, pp. 158–172, 1992.

[147] J. Shanteau, D. J. Weiss, R. P. Thomas, and J. C. Pounds, "Performance-based assessment of expertise: How to decide if someone is an expert or not," *European Journal of Operational Research*, vol. 136, no. 2, pp. 253–263, 2002.

[148] R. M. Cooke, *Experts in Uncertainty: Opinions and Subjective Probability in Science*. New York, New York, USA: Open University Press, 1991.

[149] R. French, "The Turing Test: the first 50 years.," *Trends in cognitive sciences*, vol. 4, no. 3, pp. 115-122, Mar. 2000.

[150] R. M. O'Keefe and D. E. O'Leary, "Expert system verification and validation: a survey and tutorial," *Artificial Intelligence Review*, vol. 7, no. 1, pp. 3-42, Feb. 1993.

[151] R. Agarwal, R. Kannan, and M. Tanniru, "Formal validation of a knowledge-based system using a variation of the Turing test," *Expert Systems with Applications*, vol. 6, no. 2, pp. 181-192, Apr. 1993.

Part two: Papers

Paper A: A probabilistic relational model for security risk analysis

Teodor Sommestad, Mathias Ekstedt and Pontus Johnson

Abstract

Information system security risk, defined as the product of the monetary losses associated with security incidents and the probability that they occur, is a suitable decision criterion when considering different information system architectures. This paper describes how probabilistic relational models can be used to specify architecture metamodels so that security risk can be inferred from metamodel-instantiations.

A probabilistic relational model contains classes, attributes, and class-relationships. It can be used to specify architectural metamodels similar to class diagrams in the Unified Modeling Language. In addition, a probabilistic relational model makes it possible to associate a probabilistic dependency model to the attributes of classes in the architectural metamodel. This paper proposes a set of abstract classes that can be used to create probabilistic relational models so that they enable inference of security risk from instantiated architecture models. If an architecture metamodel is created by specializing the abstract classes proposed in this paper, the instantiations of the metamodel will generate a probabilistic dependency model that can be used to calculate the security risk associated with these instantiations. The abstract classes make it possible to derive the dependency model and calculate security risk from an instance model that only specifies assets and their relationships to each other. Hence, the person instantiating the architecture metamodel is not required to assess complex security attributes to quantify security risk using the instance model.

1 Introduction

Security issues related to information technology continue to be a concern in today's society, and for decision makers in it. Security is a complex property, and several diverse factors need to be considered to assess the security of a system's architecture. To support decision makers a plethora of approaches, frameworks and methods has been proposed for analyzing and ranking security – all with some explicit or implicit definition of security.

From a decision maker's perspective, tools and techniques to assess security of both existing and potential future architectures are needed. There is also a need to relate the result of such an assessment to business decisions, such as investment alternatives that strengthen security. The concept of risk, defined as the product of the monetary losses associated with security incidents and the probability that they occur, has been suggested as a suitable input to decision making [1,2]. Several financial methods with risk measurements as a basis have also been adapted for security to provide decision makers with tools to manage security efficiently from a business perspective. For example return on security investment [3], and the methods presented in [4-7].

Although risk is well defined and practical for decision making, it is often difficult to calculate a priori. Analysis frameworks such as [4] restrict themselves to three variables: the probability that a threat surfaces, the probability that an attack succeeds, and the loss suffered from a successful attack. While quantifying these variables provides the necessary means for assessing risk, it is not apparent how to obtain the numbers needed to do so. Decision makers typically have an understanding of the architecture of their organization and its systems. However, their understanding of the dependencies among the properties of risk treatments, the threat environment and sensitive assets is hazy. Methods that support decision makers by deriving security risk associated with both existing and potential future architectures are thus desirable.

Architectural models provide decision makers with a convenient tool to abstract and capture different aspects of information systems in diagrammatic descriptions. Metamodels like the one offered in CORAS [8] guide the modeler to create graphical descriptions that can be used to assess risk. This type of metamodels does however not help the modeler to identify the risks which their particular architecture face, and do not provide the data needed to quantify security or risk based on the model. This analysis is instead left for the user of the metamodel. Methods such as [9] generate attack graphs from descriptions of computer networks and offer an alternative when the decision concerns network security. But these do not provide support for assessing the probability that a certain threat surfaces, i.e. that certain attack steps are attempted, nor do they include losses in the models. Consequently, they do not produce a measure of security risk for the decision maker. This paper describes a formalism for constructing architecture metamodels so that security risk can be inferred from the metamodel's instantiations.

1.1 Architecture models and security risk analysis

If security risk could be easily quantified from architecture models of information systems this would provide an intuitive way to assess the security risk associated with both the current "as-is" scenario, and potential future "to-be" scenarios. The decision maker would create models of different architectures by representing relevant objects and relationships in diagrammatic descriptions and from these assess the security risk associated with the architectures. These architecture models may cover management aspects, operational aspects or pure technical aspects. They can for instance be created to assess the security risk associated with different network architectures, or to assess the impact of different password policies on the overall security risk.

To make accurate predictions from an architecture model it needs to represent objects and relationships that influence security risk. If network architectures are assessed, it would for example be of relevance to include information on the placement of firewalls in the architecture model. A metamodel can guide the decision maker to create instance models that include relevant objects and relationships. To provide this guidance the metamodel must resolve how security risks (according to some theory) depend on different architectures, at least to some level of detail.

If the security risk was possible to compute based on the theory of how risk relates to different architectures it would relieve the decision maker of extensive analytical efforts. Security risk could then be derived from instantiated architecture models, and the decision maker would only be required to represent the objects and relationships that constitute the architecture.

This paper proposes the use of probabilistic relational models (PRMs) [10] to specify metamodels for security risk analysis. A PRM is similar to a Class Diagram in the Unified Modeling Language (UML) [11] and contains classes, attributes, and class-relationships. In addition, a PRM makes it possible to associate a probabilistic model to the metamodel by defining relationships between the attributes of classes in the metamodel. More specifically, a PRM makes it possible to define how the value of one attribute depends on the value of other attributes in an architectural model. With these elements a PRM allows, in a general sense, architecture metamodels to be coupled to a probabilistic inference engine. A PRM can for instance specify how different logical network architectures and properties of its users influence the security risk an organization faces. Hence, if metamodels are expressed using the PRM formalism it can be specified how security risk should be inferred from the metamodel's instantiations.

There is however an infinite number of (more or less suitable) ways that a PRM can be structured for security risk analysis. A number of concepts need to be related to each other when security risk is assessed. The main contribution of this paper is the proposition of a package of abstract PRM-classes that can be used to create PRMs that infer security risk from architecture models.

The proposed class-package is expressed as a PRM and specifies a set of classes, attributes, class-relationships and a probabilistic model for how attributes of these classes depend on each other.

The classes in this PRM are abstract and cannot be directly instantiated into an architecture model. They can however be made concrete if they are specialized into subclasses according to a set of constraints. If architecture models are instantiations of such concrete classes, then security risk is possible to infer from the architecture model. This inference can also be performed on architecture models that merely represent assets and assets relationships to each other. Hence, little security expertise is required to instantiate the architecture model, and security risk can still be inferred.

1.2 Outline

Chapter two describes related work within the field of security and risk analysis. Chapter three explains the PRM formalism and the terminology associated with it. Chapter four describes the relationship between the different models presented in subsequent chapters. Chapter five presents the main contribution of this paper – a PRM consisting of abstract classes that are associated with a set of constraints that state how these can be specialized into concrete subclasses. Chapter six exemplifies how these abstract classes can be specialized into concrete classes and how probabilistic models can be associated with these classes. In chapter seven a case study applying these specialized (concrete) classes to assess security risk associated with an automation system in power station is described. In chapter eight the proposed modeling method is discussed, and in chapter nine conclusions are drawn.

2 Related works

The use of architectural modeling languages has a long history in management and development of information systems. Modeling languages such as the Unified Modeling Language (UML) [11], the Systems Modeling Language (SySML) [12], and the Business Process Modeling Notation (BPMN) [13] provide support to create diagrammatic descriptions of information system architectures and system environments. These diagrammatic architecture descriptions can be developed for a variety of purposes, including different types of analysis. One aspect that can be analyzed based on architectural descriptions is security, and there are several methods and modeling languages specifically supporting this purpose. The formalism presented in this paper supports quantification of security risk based on system architecture models, and does not require security expertise to perform the actual quantification. This section will explain how similar modeling languages and methods relate to the one proposed in this paper.

There are several modeling languages targeted at providing support for security assessments in early phases of system development. Methods like misuse cases [14] and abuse cases [15, 16] align with well established methods for software requirements engineering to provide support for depicting potential threats and use cases that mitigate these. While these two offers a language to describe threats and countermeasures, they are not associated with methods to quantitatively assess security from the models created. This is also the case for languages such as SecureUML [17] and SPML [18] that use models with the purpose of model-driven development.

Two other modeling languages that are intended for the system development phases are Secure Tropos [19] and UMLsec [20]. Both of these provide a language and methodology which can provide a basis for security assessments. Secure Tropos extends the Tropos methodology [52] and can be used to specify security concerns associated with planned systems. The UML extension UMLsec provides a language to depict security-relevant information in diagrams describing a system specification [20]. Both these also provide support for automated verification of architecture models. Secure Tropos is associated with a set of rules that can be used to verify if goals related to trust are fulfilled when trust is delegated [50]. UMLsec makes it possible to evaluate if UMLsec-diagram fulfill a set of stipulated requirements [20,51]. The output of these automated analysis methods are however a pass/fail result that state if the architecture fulfill the requirements. Such verifications can support security risk analysis, but there are no automated means to compute security risk directly from them.

The pass/fail output is also a characteristic of the Common Criteria (CC) [21]. The CC framework offers a method to specify security requirements and evaluate their fulfillment. In CC's general conceptual model the relationships between owners, assets, risks, countermeasures, threat agents and threats are described. However, an evaluation employing CC's framework does not quantify risk. Instead it provides a pass/fail result together with a rating of the assurance level (1-7) of the evaluation.

A method specifically developed for analyzing and quantifying risk is CORAS [8]. With guidance from CORAS's metamodel, a graphical description of the threat scenario is created and used as a support to determine if, and how, the identified risks should be treated. This is done by modeling the relationships between assets, threats, vulnerabilities, unwanted events, risks and treatments. Although risk in CORAS is defined as the product of likelihood and consequence, there is no analysis framework coupled to the metamodel and thus no algorithmic method to calculate risk based on a graphical description. There is also no description of what different types of risk treatments that should be modeled, or how risk treatments influence risks in CORAS. These calculations, as well as the content of the CORAS diagram, must instead be assessed by the persons applying CORAS.

Several other analysis methods also depend on analysts to quantify risk. CCTA Risk Analysis and Management Method (CRAMM) [22] offer a structured method to assess risk qualitatively by identifying: 1) how frequent an incident occurs, 2) the probability that incidents would result in a worst case scenario, and 3) loss values. These three values are used to produce a monetary value for annual loss expectancy. Information Security Risk Analysis Method (ISRAM) [23] does in a similar way guide the analyst to assess probabilities for security incidents to occur and to assess the potential consequences of these. The same type of guidance is also provided by Operationally Critical Threat, Asset and Vulnerability Evaluation (OCTAVE) [24].

Threat trees [25] and attack trees [26,27] are graphical notations that have evolved from fault trees, used to illustrate attackers' goals together with possible ways to reach these goals. The attacker's main goal is depicted as the root of the tree and the steps to reach this goal are broken down into sub-goals of the attack through "AND" and "OR" relationships. Threat trees and attack trees have been applied in several ways to assess security. In [25] it is suggested that the threat trees should be used to rank the threats is terms of risk. In [27] it is suggested that attack trees can be used to assess if attacks are possible, if special tools are needed or how much effort an attack requires.

A similar method for representing attacks is attack graphs. In attack graphs the sequence of steps needed to accomplish the attack is expressed rather than the set of steps; and this is modeled in a graph structure instead of a tree structure. Attack graphs have for instance been used to assess the probability that an attacker reaches particular attack step [28], or to analyze the security of system configurations in terms of the weakest adversary that can compromise the network [29]. Several similar analysis methods exist in addition to these (see for example [30-33]). A problem with attack graphs is that they scale poorly and become extremely complex even for moderately sized system architectures. Methods have been proposed to reduce the complexity and computational problems associated with attack graphs in [30,31,34]. Probabilistic treatment of the relationship between different attack steps is another suggested solution to the scalability problem. In [35] Bayesian networks are used to represent attack graphs more compactly and to calculate the probability that a network-attack succeeds, as oppose to doing so deterministically.

The methods based on trees and graphs provide a link between vulnerabilities and plausible consequences. Their structure also facilitates straightforward means for analysis, for example of how likely an how likely an attack is to succeed or if an attack step is reachable. None of these abovementioned methods does however offer means for assessing how likely an adversary is to attempt combinations of attacks steps. Hence, the probability of a certain attack being realized cannot be inferred and security risk cannot be calculated.

A natural extension of attack trees and attack graphs is to include controllable countermeasures in the model. In [25] countermeasures are added as leaves of threat trees; in [36] and [37] it is shown how threats and countermeasures can be related to each other in tree structures; in [26] countermeasures and attacks are connected in directed acyclic graphs. In [38] attack trees with countermeasures as leaves are called Defense trees. Techniques have been presented which use Defense trees to model strategic games in security [39], to model conditional preference of defense techniques using conditional preference nets [40], and for performing economic evaluation of security investments [38].

The method described in [38] allows a modeler to specify a defense tree and a number of variable values. From this, return on security investment as well as the adversary's return on attack can be inferred. As a step in these calculations annual loss expectancy is derived. There is however no method for deriving the defense tree or the variable values required from an architectural description, and new variable values must be defined after an option is chosen. Hence, the method relies on analysts to continuously update the model.

Methods have been developed to ease the burden of creating tree and graph structures by deriving them from architectural models. In [41-43] methods are described for deriving attack graphs from network configurations and known vulnerabilities; in [44] a method for deriving probabilistic defense graphs from architectural models is described. These methods do however focus strictly on prevention of attempted attacks and lack the power to represent treatments that limit the losses from successful attacks or deter adversaries from attempting them. Consequently, they do not provide the information required to determine security risk. The methods presented in [41-43] are also focused entirely on computer networks and thus lack the capability to represent other facets of security, such as human behavior or organizational policies.

Unlike the abovementioned approaches the formalism presented herein makes it possible to specify how expected loss should be quantified from an architecture model. The formalism makes it possible to specify architectural metamodels that generates a probabilistic dependency model when they are instantiated. The conceptual structure of this dependency model extends the conceptual model presented in Common Criteria [21] by defining attack steps as a part of a threat. Attack steps are related to countermeasures in directed acyclic graphs, similar to the graphs presented in [26]. Attack steps can also be associated to each other probabilistically, similar to the probabilistic attack graphs in [35]. With this extension the proposed dependency model allows inference of how probable attempts are and how probable attempts are to succeed, as well as inference of expected loss.

3 Probabilistic relational models

A *probabilistic relational model (PRM)* [10] specifies a template for a probability distribution over an architecture model. The template describes the metamodel for the architecture model, and the probabilistic dependencies between attributes of the architecture objects. A PRM, together with an instantiated architecture model of specific objects and relations, defines probability distributions over the attributes of the objects. The probability distributions are then used to infer the values of unknown attributes.

3.1 Architecture metamodel

An architecture metamodel M describes a set of classes, $X_1,\ldots,X_n$. Each class is associated with a set of descriptive attributes and a set of reference slots (relationships).

The set of descriptive attributes of a class X is denoted $A(X)$. Attribute A of class X is denoted $X.A$ and its domain of values is denoted $V(X.A)$. For example, in Figure 1, the class *System* has the two descriptive attributes *Availability* and *Reliability*, both with the domain {*Low, Medium, High*}.

The set of reference slots of a class X is denoted $R(X)$. We use $X.\varphi$ to denote the reference slot φ of X. Each reference slot φ is typed with the domain type $Dom[\varphi]= X_i$ and the range type $Range[\varphi]= X_j$. A reference slot φ denotes a function from X_i to X_j, and its inverse φ^{-1} denotes a function from X_j to X_i. The class *SystemAdministrator* in Figure 1 has the reference slot *Administrates* whose range is the class *System*. The reference slot *System.Administrates*$^{-1}$ then has range *SystemAdministrator*. Thus, the fundamental modeling constructs of PRMs are the same as in general conceptual modeling techniques.

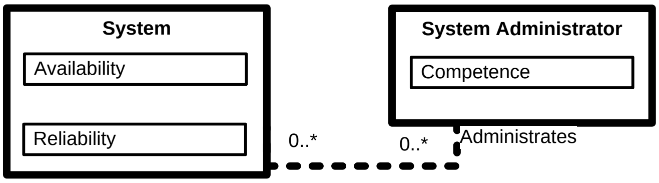


Figure 1. An example meta-model.

3.2 Architecture instantiation

An architecture *instantiation I* (i.e. an architecture model) specifies the set of objects in each class *X*, and the values for attribute(s) and reference slot(s) of each object. For example, Figure 2 presents an instantiation of the meta-model described in Figure 2. It specifies a two particular *System*-object, particular *SystemAdministrator*, and the references between these. Values to the attributes are not yet ascribed.

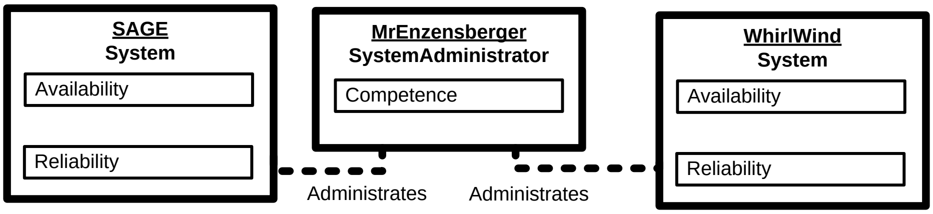


Figure 2. An example relational skeleton, i.e. a partly instantiated model.

3.3 Probabilistic model over attributes

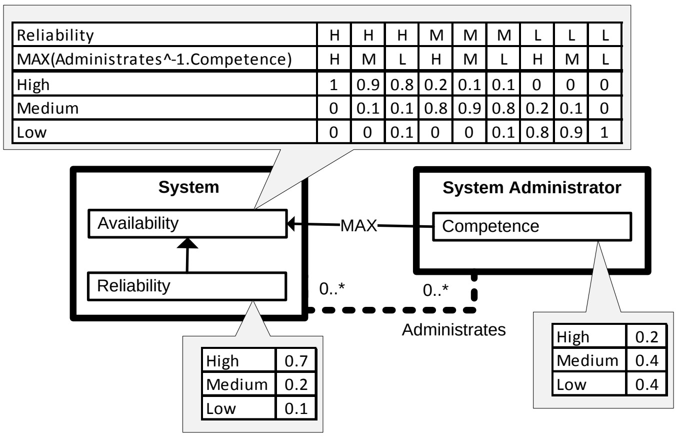
So far, the probabilistic relations between the attributes have not been addressed. A PRM *Π* specifies a probability distribution over all instantiations *I* of the metamodel *M*. This distribution is expressed in terms of a Bayesian network [45] and it consists of a qualitative dependency structure, and associated quantitative parameters.

The qualitative dependency structure is defined by associating attributes *X.A* with a set of parents *Pa(X.A)* that causally influence these attributes. Each parent of *X.A* has the form *X.τ.B* where $B \in A(X.\tau)$ and *τ* is either empty, a single reference slot *φ* or a sequence of reference slots $\varphi_1,\ldots,\varphi_k$ such that for all *i*,

$Range[\varphi_i]=Dom[\varphi_{(i+1)}]$. We call τ a slot chain. Note that when $X.\tau.B$ reference attributes external to the class X, it might be referencing a set of attributes rather than a single one since there may exist multiple instantiated objects of one class. In these cases, $X.A$ depends probabilistically on an aggregation function over those attributes. In general these aggregation functions could take any form. In this paper however, the logical operations *AND*, *OR*, and the arithmetic operation *SUM* (for summing).

In Figure 3 the running example is extended with two attribute dependencies. Firstly, the attribute *Availability* of the class *System* have the attribute *Reliability* of the same class as parent, meaning that the external property availability is dependent on the more internal property reliability. Secondly, *System.Availability* is also dependent on the attribute $System.Administrates^{-1}.Competence$, i.e. the competence of the system administrators that administrates the system. In the example this dependency is aggregated in terms of MAX function essentially illustrating that the system availability will be dependent on the most competent administrator. These aggregate properties are associated with a probability distribution for the case when $X.\tau.B$ is an empty set in the architecture instantiation. For instance, if $MAX(Administrates^{-1}.Competence)$could return *Low* if there is no administrator assigned.

Given a set of parents for an attribute we can now define a probability model by associating a conditional probability distribution with the attribute, $P(X.A\ |Pa(X.A))$. For instance, $P(System.Availability=High|\ Reliability=High,\ MAX(Administrates^{-1}.Competence)=Medium)=90\%$ specifies the probability that the system has a high availability given that the most competent system administrator has a medium competence and the system has a high reliability. Conditional probability distribution tables for the running example are presented in Figure 3.



Reliability	H	H	H	M	M	M	L	L	L
MAX(Administrates^-1.Competence)	H	M	L	H	M	L	H	M	L
High	1	0.9	0.8	0.2	0.1	0.1	0	0	0
Medium	0	0.1	0.1	0.8	0.9	0.8	0.2	0.1	0
Low	0	0	0.1	0	0	0.1	0.8	0.9	1

High	0.7
Medium	0.2
Low	0.1

High	0.2
Medium	0.4
Low	0.4

Figure 3. An example meta-model including a qualitative and a quantitative dependency structure.

We can now define a PRM *Π* for a meta-model *M* as follows. For each class *X* and each descriptive attribute *A* ∈ *A(X)*, we have a set of parents *Pa(X.A)*, and a conditional probability distribution that represents $P_{\Pi}(X.A|Pa(X.A))$.

Given an instantiated metamodel without attribute values, σ_r, a PRM *Π* specifies a probability distribution over a set of instantiations *I* consistent with σ_r:

$$\boldsymbol{P(\mathcal{J}|\sigma_r, \Pi) = \prod_{x\in\sigma_r(X)} \prod_{A\in At(x)} P(x.A|Pa(x.A))}$$

where $\sigma_r(X)$ are the objects of each class in the instantiated metamodel. Hence, the attribute values can be inferred.

A PRM thus constitutes a formal machinery for calculating the probabilities of various architecture instantiations. This allows us to infer the probability that a certain attribute assumes a specific value, given some (possibly incomplete) evidence of the rest of the architecture instantiation.

In essence, a PRM define how a Bayesian network shall be generated over the attributes in an instance model. An extension of Bayesian networks intended to support decision-making is so called influence diagrams [47]. Influence diagrams include attributes that represents utility which are sought to be maximized or minimized. These utility nodes have a domain of utility values which are just as regular Bayesian attribute values also ascribed probabilities. It is shown in [46] that PRMs can easily be extended to also include attributes representing the utility nodes used in influence diagrams [47]. In this paper a PRM with this extension is used.

3.4 Class inheritance and class specialization

PRMs further allow specializing classes through inheritance relationships. Classes can be related to each other using the subclass relation $\ll$. If *ERPSystem* $\ll$ *System* then *ERPSystem* is a subclass of *System* and *System* is a superclass of *ERPSystem*. Let the finite set of subclasses to class X be $C[X]$. So if $Z, Y \in C[X]$, both Z and Y are subclasses of X. A subclass Y always contains the reference slots and attributes of its superclass X. As exemplified in Figure 4, *At(ERPSystem)* is a subset of *At(System)* and *R(ERPSystem)* is a subset of *R(System)*. The conditional probability distributions of inherited attributes can however be specialized in subclasses. A subclass can also refine the range of inherited reference slots.

For each subclass $Y \in C[X]$ and inherited attribute $B \in A(X)$, there are parents $Pa(Y.B)$ and a conditional probability distribution $P(Y.B \mid Pa(Y.B))$. These can be equal to attributes and parents in the superclass X, i.e. $Pa(Y.B)=Pa(X.B)$ and $P(Y.B \mid Pa(Y.B))= P(X.B \mid Pa(X.B))$, or they can be specialized to include additional parents in $Pa(Y.B)$ or a different conditional probability distribution for $P(Y.B \mid Pa(Y.B))$. For example, as *ERPSystem* $\in$ *C[System]*, the probability distribution for *ERPSystem.Availability* may be different (specialized) from *System.Availability*. The parents of *ERPSystem.Availability* may also be different from those of *System.Availability*. If the set of parents of an attribute is changed, i.e. $Pa(Y.B)\neq Pa(X.B)$, then the probability distribution must be specialized.

A subclass $Y \in C[X]$ may also be specialized with regard to the range of its reference slots. The reference slot is then refined. Let $X.\varphi$ be a reference slot where $Range(X.\varphi)=U$. The reference slot of the subclass, $Y.\varphi$, can have the same range as $X.\varphi$, i.e. $Range(Y.\varphi)=Range(X.\varphi)=U$. Or the reference slot of the subclass, $Y.\varphi$, can be specialized by restricting it to subclasses of U, i.e. $Range(Y.\varphi)=W$, where $W \in C[U]$. For instance, if

EnterpriseSystemAdministrator is a subclass of *SystemAdministrator*, then the reference slot *EnterpriseSystemAdmin.Administrate* could be refined to the range *ERPSystem* since it is a subclass of *System*.

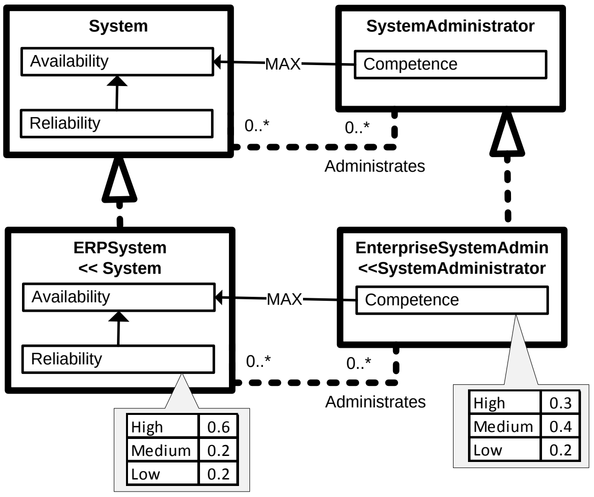


Figure 4. Subclasses of System and SystemAdministrator. The conditional probabilitiesof ERPSystem.Reliability and EnterpriseSystemAdmin.Competence are spececialized. The reference slot EnterpriseSystemAdmin.Administrates is refined to the range ERPSystem.

4 Abstract and Concrete PRM packages

This paper presents a PRM that enables calculation of the expected loss, e.g. monetary loss, due to poor security. A set of abstract classes and an incomplete probabilistic model of how attributes of these classes depend on each other is described. This set of abstract classes will in this paper be referred to as the *AbstractPRM-package* (analogous to packages in UML) and they define a structure that is favorable to metamodels supporting security risk analysis. The AbstractPRM-package is similar to the general conceptual model in CC [21] and does for example include the classes *Asset, Countermeasure, ThreatAgent* and *Threat.* By specializing these abstract classes into concrete subclasses a metamodel associated with a probabilistic model for security risk can be created (cf. Figure 5). In this paper the set of concrete classes is referred to as the *ConcretePRM-package*.

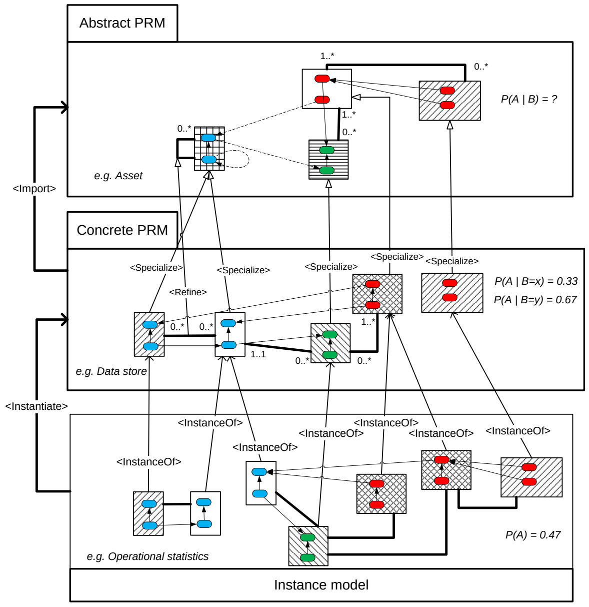


Figure 5. The AbstractPRM-package details classes, attributes and reference slots and an incomplete probabilistic model, here represented by dashed arcs. The ConcretePRM-package specialize the classes in the AbstractPRM-package and details the probabilistic model. When the ConcretePRM is instantiated a probabilistic model over security risk can be derived.

Concrete subclasses to those in the AbstractPRM package are created using inheritance relationships. The concrete class then specialize the abstract class. A ConcretePRM-package can for instance include the countermeasure *Firewall*, the threat agent *Outsider* or the asset *DataStore*. The concrete subclasses can, as described in section 3.4, both refine the reference slots of its superclass and specialize the probabilistic model of its inherited attributes. These two features are used to create a concrete class in a ConcretePRM-package – no other operations are needed. Hence, there is no need to define new attributes of classes, and no need to identify classes that are not subclasses to these in the AbstractPRM-package. The AbstractPRM-package is associated with as set of constraints that define how the probabilistic model can be specified ConcretePRM-packages. These constraints do for instance say that the attribute *PreventiveCountermeaure.Functioning* only can be a parent of *AttackStep.PossibleToAccomplish* and that the slot chain associated with this dependency should contain reference slots that connect two assets.

If these constraints are followed, instantiations of the ConcretePRM will express probabilistic models that facilitate straightforward analysis of security risk. More specifically, the constraints ensure that if the assets and asset-to-asset relationships in a ConcretePRM are instantiated, a probabilistic dependency model can be derived. This probabilistic dependency model will express the relationships between assets, countermeasures, attack steps, threats, and threat agents in the instance model. Hence, just as abstract classes in object-oriented programming languages help a developer with blueprints and core functionality, this AbstractPRM-package helps a metamodeler to create a ConcretePRM-package for security risk analysis. It ensures that instantiations of the classes in a ConcretePRM-package produce a probabilistic dependency model that can be used to infer the probability that attacks are successful and the probability that they will be attempted. It also ensures that this can be inferred from an architectural model that only describes assets and the relationships between assets. Loss values for a successful attack can either be defined in the ConcretePRM-package, or inserted into the instance model directly. Adding such values provides the necessary means to assess expected loss.

5 An AbstractPRM-package for security risk Analysis

This chapter describes the AbstractPRM-package for security risk analysis with classes, attributes, reference slots, and attribute-dependencies. This chapter is the locus of this paper's contribution. The AbstractPRM-package consist of an architectural metamodel and a set of constraints that define how its probabilistic model over its classes may be specialized in concrete subclasses. The AbstractPRM-package is depicted in Figure 6 and described below. This description is divided in two subsections. The architectural metamodel is described first; thereafter the probabilistic model is described together with the constraints that state how subclasses can be defined in ConcretePRM-packages.

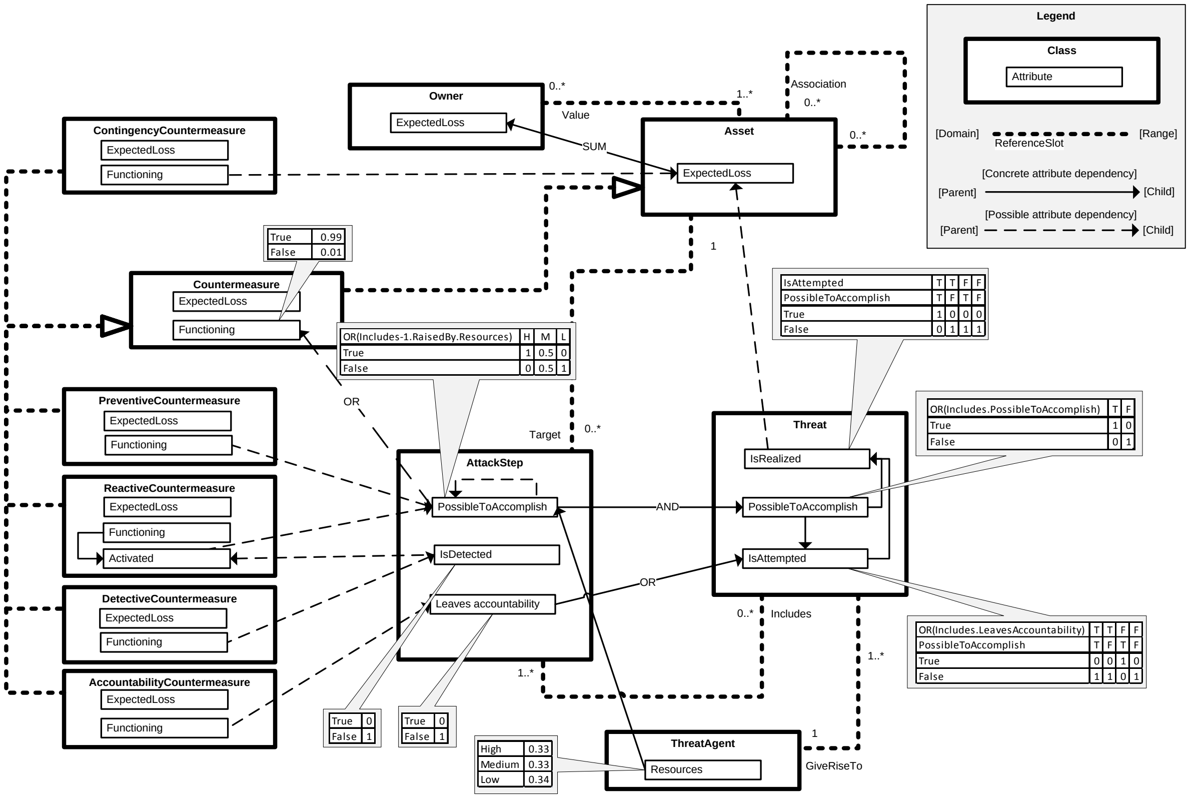


Figure 6. An AbstractPRM-package for assessing security risk. Both the architectural metamodel and probabilistic dependencies are shown here. Some probabilistic dependencies (solid arcs) are concrete, while other (dashed arcs) are potential probabilistic dependencies that may be defined in ConcretePRM-packages. Conditional probability tables describe attribute dependencies for those attribute dependencies that are concrete in this package.

Throughout the description of the AbstractPRM-package it will be referred to concepts used in Common Criteria (CC) [21]. CC defines an internationally well established terminology for security risk assessments and its general conceptual model describes concepts that relate to security risk. In CC's general conceptual model, threat agents wish to damage and/or abuse assets. Threat agents therefore give rise to threats that are associated with an asset. Threat also increase risk to assets. Owners value assets and wish to minimize risk and they can impose countermeasures to do so. The AbstractPRM-package includes these concepts in and arranges them in a similar manner as CC's general conceptual model. However, to support quantitative risk analysis some additional constructs are included in the AbstractPRM-package: attack steps are added and related to each other, threats, assets and countermeasures; classes are given descriptive attributes; and a dependency structure is defined over these attributes. The AbstractPRM and CC's general conceptual model thus differ somewhat. These differences are just as the similarities described using CC's terminology.

5.1 Architectural metamodel

As CC's general conceptual model this metamodel relates a *Threat* to the *ThreatAgent* that gives rise to the threat. This is done with the reference slot *ThreatAgent.GiveRiseTo* with range *Threat.* Unlike CCs conceptual model however, this metamodel does not directly relate a *Threat* to an *Asset.* This is instead done through the class *AttackStep*. The AbstractPRM in Figure 6 requires a set of *AttackStep*-classes to be detailed as a part of the *Threat* using the reference slot *Threat.Includes.* This set of *AttackStep*-classes is similar to the concept "attack" that in CC's shall be used together with threat agent and asset to define a threat when the "Security Environment" is described. CC's conceptual model state that a threat agent wishes to "abuse and/or damage" assets. In the AbstractPRM-package this is represented through the reference chain *ThreatAgent.GiveRiseTo.Includes.Target*, which has the range *Asset.* Hence, each *AttackStep* is associated to the *Asset* it targets using the reference slot *AttackStep.Target.*

An *AttackStep* and *Threat* can either be possible or impossible to accomplish, hence the attributes *AttackStep.PossibleToAccomplish* and *Threat.PossibleToAccomplish* exist. A *Threat* does in addition hold the attribute *IsAttempted* and *IsRealized. Threat.IsAttempted* indicates if it is attempted by the threat agent or not; *Threat.IsRealized* indicates if it is realized or not. The attribute *AttackStep.IsDetected* indicates if an ongoing attack will be detected, and *AttackStep.LeavesAccountability* says if the attack step will lead to accountability, i.e. if the threat agent can be held accountable for attempting it.

CC suggests that four aspects should be used to describe threat agents. *ThreatAgent.Resources* is an aggregate of two of these – “skills” and “resources”. CC also suggests that “motivation” and “opportunity” could be used in addition to these. The threat agent’s motivation can in this metamodel be expressed in terms of the attribute *Threat.IsAttempted.* A motivating threat will, ceteris paribus, have a higher probability of being attempted than a threat that is not motivating. Opportunity is captured by the attribute *Threat.PossibleToAccomplish,* which express the probability that an attacker can realize the threat if this is attempted.

CC’s general conceptual model includes the concept of countermeasures but does not differentiate among these with regard to how they causally reduce the risk. Outside of its conceptual model CC state that these can be seen as “Security Objectives” which are achieved by meeting “Security Functional Requirements”. The AbstractPRM-package defines the class *Countermeasure* and further specialize this class to enable more detailed quantification how countermeasures depend on each other and how they influence risk. Five subclasses of *Countermeasure* are defined: *PreventiveCountermeasure* (e.g. firewall)*, DetectiveCountermeasure* (e.g. intrusion detection system), *ReactiveCountermeasure* (e.g. incident handling)*, ContingencyCountermeasure* (e.g. backups) and *AccountabilityCountermeasure* (e.g. logging). In relation to CC’s terminology these five types of countermeasures correspond to classes of “Security Functional Requirements”. How these casually reduce risk is described by the PRM’s probabilistic dependency structure (cf. section 5.2).

Countermeasures can hold a value, and the class *Countermeasure* is therefore a subclass of *Asset*, i.e. *Countermeasure* « *Asset.* In addition to the inherited attribute *ExpectedLoss* which states value of the *Countermeasure*, *Countermeasure* and its subclasses also have the attribute *Functioning.* The attribute *Countermeasure.Functioning* expresses if the *Countermeasure* is working as it should – the countermeasure’s correctness in CC’s terminology. If a reactive countermeasure is functioning it can respond to detected events. Therefore *ReactiveCoutermeasure* also holds the attribute

ReactiveCoutermeasure.Activated which indicates if it is put into effect or not.

Just as in CC, an *Owner* value *Asset*s through the reference slot *Owner.Value*. An *Asset* can in this model also be related to other assets using the reference slot *Asset.Association*. The risk to assets and owners is in the architecture metamodel represented by *Asset.ExpectedLoss* and *Owner.ExpectedLoss*. Both *V(Asset.ExpectedLoss)* and *V(Owner.ExpectedLoss*) are scalars that express losses, e.g. monetary losses. The attribute *ThreatAgent.Resources* has the domain of values {*High, Medium, Low*} whose semantics is defined in ConcretePRM-packages. All other attributes in the metamodel have the domain of values {*True, False*}.

5.2 Probabilistic dependency structure and specialization constraints

As CC's conceptual model, the PRM depicted in Figure 6 contains concepts that influence security risk. This metamodel is in addition associated with a dependency structure which defines how the attributes of these concepts influence each other. To properly define a PRM it must be described how to derive the parents *Pa(X.A)* of all attributes *X.A* in an instantiated model. Attributes also need to be associated with a conditional probability distribution *P(X.A|Pa(X.A))*.

Figure 6 shows the probabilistic dependency structure as solid and dashed arcs spanning between attributes. An arc is solid when the abstract package defines how parent-attributes should be derived from an instance model, and dashed when it does not. Hence, the slot chains that define dashed arcs must be specified in packages that import the AbstractPRM-package, i.e. in ConcretePRM-packages.

Parents are left undefined when the AbstractPRM-package's metamodel does not include the information required to determine them. For example, a data backup for a data store would be a typical *ContingencyCountermeasure* in an instance model, and a data store would be a typical *Asset*. The dashed arc from

ContingencyCountermeasure.Functioning to *Asset.ExpectedLoss* then implies that a functioning data backups influence the expected losses of the data store. This dependency structure would be reasonable if a model would instantiate an *Asset* that is a data store, this *Asset* is associated with an instance of *ContingencyCountermeasure* represent the data store's data backup. But, if the asset instead is an employee, a data backup is of little help. There is however nothing in the AbstractPRM-package that allows instance models to represent if the asset is a data store, if the countermeasure is a backup-system, or if the backup-system makes backups of the data store.

The concepts necessary to specify theories like these can however be specified in ConcretePRM packages that specialize the classes in the AbstractPRM-package. To do this a set of asset-to-asset references (*Asset.Association*) should be used. Chapter 6 shows an example of how this can be done in a ConcretePRM-package. This ConcretePRM-package does for instance say that the class *Backup* is a subclass of *ContingencyCountermeasure* and that the class *DataStore* is as a subclass of *Asset*. If these two are related with the asset-to-asset reference *DataStore.Has* it specifies that *Backup.Functioning* is a parent of *DataStore.ExpectedLoss*, i.e. that *DataStore.ExpectedLoss* has parents *DataStore.Has.Functioning*.

Asset-to-asset references are used to define the slot chains that make dashed arcs concrete (solid). For the rules dictating how the attribute-parents can be specified, recall that parents to an attribute $X.A$ are defined in the form $X.\tau.B$, where $B \in A(X.\tau)$. The slot chain τ is here either empty, a single slot ϱ or a sequence of slots $\varrho_1,\dots,\varrho_k$ such that for all i, $Range[\varrho_i\ ]=Dom[\varrho_{(i+1)}]$. Also recall that this AbstractPRM-package states that *Asset* can have zero to many references to other *Asset*s through the reference slot *Association*. Concrete specializations of the classes in the AbstractPRM-package can use refinements of the reference slot *Association* to specify concrete attribute dependencies. A refinement could for example be the reference slot *DataStore.Has*, as in the example above.

Table 1 describes the parents of attributes in the AbstractPRM. Here *a2a* is used to represent a chain of refined asset-to-asset relationships, i.e. a chain of refined *Association* reference slots.

The range of the reference chain is in some cases constrained to a specific subclass of asset. If this is the case the class' name is shown within parentheses at the end of the chain. Table 1 does for example show that *Asset.ExpectedLoss* can have the parent *Asset.a2a.(ContingencyCountermeasure).Functioning*. In a ConcretePRM-package this can be used to associate a *DataStore.ExpectedLoss* with a contingency countermeasure of type *Backup*, as in the example above. The slot chain *a2a* would in this case be the single reference *Has*, thus making *DataStore.Has.Functioning* a parent of *DataStore.ExpectedLoss*. More formally, if <*AssociationRefinement*> denotes a refinement of the reference slot *Asset.Association*, then a chain of <*AssociationRefinement*> reference slots is denoted as *a2a*.

This parent structure is defined to infer a value for *Owner.ExpectedLoss*. This value will be the sum of expected losses of the *Assets* that the *Owner* values. *Asset.ExpectedLoss* is caused by *Threat*s that are realized [21], and *Threats.IsRealized* can therefore be defined as a parent of *Asset.ExpectedLoss*. As defined in [4], a threat will be realized (*Threat.IsRealized*) if it is attempted (*Threat.IsAttempted*) and if it is possible to succeed with (*Threat.PossibleToAccomplish*). For *Threat.PossibleToAccomplish* to be true, all *AttackSteps* included in the threat must be possible to accomplish. The attack steps included in a threat thus correspond to an attack path in an attack graph, and all these attack steps must be possible to accomplish if the attack path should be possible to accomplish.

*AttackStep*s can be associated to each other in a way similar to the probabilistic attack graphs described in [35]. It can be specified that accomplishment of one attack step influences the probability that another attack step will be possible to accomplish. This is specified using asset-to-asset relationships in the metamodel. The probability that the included *AttackStep*s are accomplished also depends on the *Resources* of the *ThreatAgent* [21]. The attribute *Threat.IsAttempted* is influenced by deterrent factors [26]: the personal risk associated with attempting the attack and the difficult of accomplishing with it. To represent this *Threat.IsAttempted* has parents *Threat.PossibleToAccomplish* and the OR-aggregate of *LeavesAcountability*-attributes in included *AttackStep*-objects.

Table 1. Attributes in the AbstractPRM-package are here shown together with the slot chains defining their parents. Owner.ExpectedLoss does for example have the parent(s) Owner.Values.ExpectedLoss. A chain of refined reference slots of Asset.Association is here denoted a2a.

Attribute
Owner.ExpectedLoss
Owner.Value.ExpectedLoss
Asset.ExpectedLoss
Asset.a2a.Target^{-1}.Includes^{-1}.IsRealized
Asset.a2a.(ContingencyCountermeasure).Functioning
Threat.IsRealized
Threat.PossibleToAccomplish
Threat.IsAttempted
Threat.PossibleToAccomplish
Threat.Includes.PossibleToAccomplish
Threat.IsAttempted
Threat.Includes.LeavesAccountability
Threat.PossibleToAccomplish
AttackStep.PossibleToAccomplish
AttackStep.Target.a2a.Target^{-1}.PossibleToAccomplish
AttackStep.Target.a2a.(PreventiveCountermeasure).Functioning
AttackStep.Target.a2a.(ReactiveCountermeasure).Activated
AttackStep.Includes^{-1}.GiveRiseTo^{-1}.Resources
AttackStep.LeavesAccountability
AttackStep.a2a.(AccountabilityCountermeasure).Functioning
AttackStep.Detected
AttackStep.a2a.(DetectiveCountermeasure).Functioning
ReactiveCountermeasure.Activated
ReactiveCountermeasure.Functioning
ReactiveCountermeasure.a2a.Target^{-1}.Detected
Countermeasure.Functioning
Countermeasure.Target^{-1}.PossibleToAccomplish

As in [26,36,37], an *AttackStep* that targets a *Countermeasure* will do so to disable it. Hence, *Countermeasure.Functioning* has the parent *Countermeasure.Target^{-1}.PossibleToAccomplish*. Functioning countermeasures can lower *Owner.ExpectedLoss*, but depending on their class they will do so differently. The dashed arcs in Figure 6 show which potential parents there could be in ConcretePRM. One internal dependency exists among the subclasses of countermeasures. Time-based security prescribe that one way of preventing attacks are to detect them and trigger reactive measure to mitigate them [48]. *ReactiveCountermeasure.Activated* can therefore be influenced by *AttackStep.Detected*.

In this model the logical operations *AND*, *OR* and the arithmetic operation *SUM* are used to aggregate multiple parents into one value. The conditional probability distribution *Threat.IsAttempted* is for example specified for the OR-aggregate of all *Threat.Includes.LeavesAccountability*-parents (cf. Figure 6). The aggregate of the logical operations *AND* and *OR* is here defined as *False* for an empty set of parents. Also let *AttackStep.PossibleToAccomplish* be *False* if *Pa(AttackStep.PossibleToAccomplish)* ∈ *At(ThreatAgent)* is the empty set.

The attributes *AssetExpectedLoss* and *Owner.ExpecredLoss* are defined as value tables with a default value of zero. All *Countermeasure* share the same conditional distribution for the attribute *Functioning*. Figure 6 describes this conditional probability, as well as all other conditional probabilities. These may, and ought to, be specialized in concrete subclasses since more accurate tables can be defined when classes are more concrete. In concrete specializations the parents of attributes targeted by dashed arcs may change as new parents are defined. In these cases the conditional probability distribution must be specialized in the subclass.

6 A ConcretePRM-package

This chapter exemplifies how a Concrete PRM can be created using the AbstractPRM-package presented in chapter 5. This is done by defining subclasses to those that exist in the AbstractPRM, by specializing the reference slots and attributes of these subclasses, and concretizing the probabilistic model using asset-to-asset references. No operations except these three are needed to create a ConcretePRM-package.

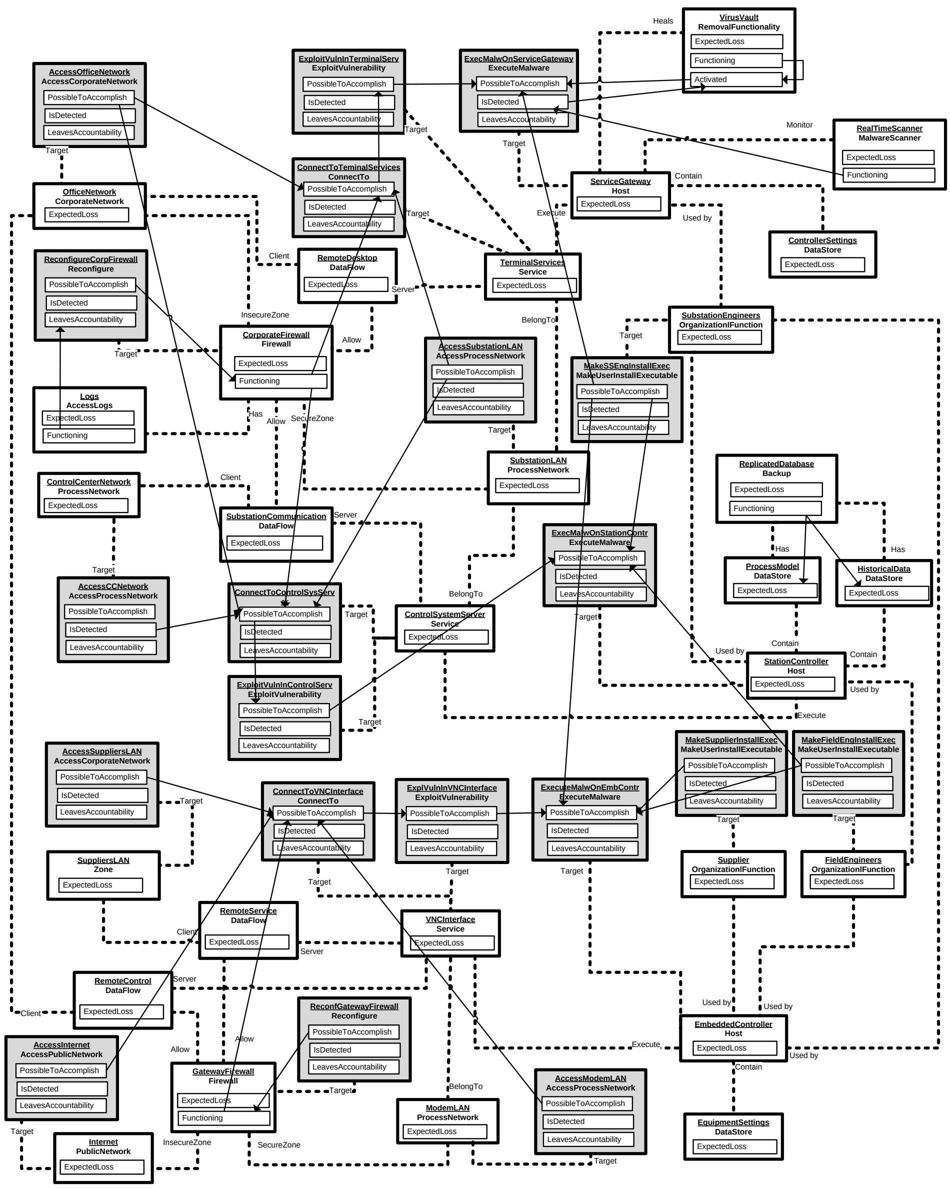


Figure 7. These concrete classes are subclasses of those depicted in Figure 6 and concretize their attribute dependencies to create a metamodel over malware attacks that cause loss of data. The conditional probabilities for some attributes are inherited from their superclass and can be found in Figure 6. Those that are specialized are shown in this figure. Loss values (marked with question marks) are detailed in the instance model.

The ConcretePRM-package described in this chapter is only an illustrative example of how a Concrete-PRM package can be constructed. Hence, its structure and conditional probabilities have not been validated. The example shows a simple metamodel that can be used to analyze risks associated with malware attacks that cause loss of data. Its description is divided in four sections, each describing a part of the Concrete-PRM package. The first three sections depict different parts of the diagram in Figure 7, namely: "Firewalls, services and network zones", "Malware and anti-malware", and "Social engineering and backup data". The fourth part describes subclasses of *Threats* and *ThreatAgents* and relates these to the classes depicted in Figure 7. The fourth part is illustrated in Figure 8.

When reference slots of type *Asset.Association* inherited from the AbstractPRM-package are refined they will be given descriptive names, e.g. *DataStore.Has*. When it is necessary to distinguish between two reference slots refinements of other types of reference slots we will use an alphabetical suffix (e.g. *X.TargetA* and *Y.TargetB*).

6.1 Firewalls, services and network zones

The primary purpose of firewalls is to control access to network addresses. They do so by blocking unwanted data flows from adjacent zones, and allow those that are wanted. With a protection scheme following the principle "deny by default", a *Firewall* will allow a number of *Data flows* to pass into the secure *Zone* from other *Zone*s. In this ConcretePRM a *Firewall* holds the reference slots *Allow, SecureZone* and *InsecureZone*, where *Range(Firewall.Allow)=DataFlow*, *Range(Firewall.SecureZone)=Zone* and *Range(Firewall.InSecureZone)=Zone*. The S*ecureZone* and *InsecureZone* here refer to the zones that are directly separated by the firewall.

A *Zone* can be targeted in the attack step *AccessZone*. Three subclasses of *Zone* are represented in the PRM: *PublicNetwork*, *CorporateNetwork* and *ProcessNetwork*. For these three the attack step *AccessZone* is specialized into *AccessPublicNetwork*, *AccessCorporateNetwork* and *AccessProcessNetwork*. This way their conditional probability distributions can be specialized.

A *DataFlow* is initiated from one *Zone* and is established to a *Service*. It has the reference slots *Client* and *Server* where *Range(DataFlow.Client)=Zone* and *Range(DataFlow.Server)* is *Zone*. A *Service* belongs to a *Zone* with the reference slot *BelongTo*, where *Range(Service.BelongTo)=Zone*.

ConnectTo is an attack step that targets a *Service*, i.e. *ConnectTo « Attack step* and *Range(ConnectTo.Target)=Service*. A *Firewall* will prevent *Services* in the secure *Zone* from unauthorized *DataFlows* where the client is in an insecure *Zone*. Hence, *Firewall « PreventiveCountermeasure* and *ConnectTo.PossibleToAccomplish* have parents *ConnectTo.TargetA.BelongTo*$^{-1}$*.SecureZone*$^{-1}$*.Functioning* and *ConnectTo.Target.BelongTo*$^{-1}$*.SecureZone*$^{-1}$*.InSecureZone.Target.PossibleToAccomplish*.

Table 2. Parents to attributes in the Firewall part of the ConcretePRM-package.

ConnectTo.PossibleToAccomplish	
	ConnectTo.TargetA.BelongTo.Target.PossibleToAccomplish
	ConnectTo.TargetA.Server^{-1}.Client.TargetB^{-1}.PossibleToAccomplish
	ConnectTo.TargetA.BelongTo.SecureZone^{-1}.InsecureZone.Target.PossibleToAccomplish
	ConnectTo.TargetA.BelongTo. SecureZone^{-1}.Functioning
Firewall.Functioning	
	Firewall.Target^{-1}.PossibleToAccomplish
Reconfigure.LeavesAccountability	
	Reconfigure.Target.Has.Functioning

A firewall will not prevent connection attempts from the zone where the service belongs, or prevent a connection attempt that uses a data flow which is allowed by the firewall. To express this *ConnectTo.PossibleToAccomplish* has the parents: *ConnectTo.TargetA.BelongTo.BelongTo*$^{-1}$*.TargetB*$^{-1}$*.PossibleToAccomplish* and *ConnectTo.TargetA.Server*$^{-1}$*.Client.Target*^{1}*.PossibleToAccomplish*.

Firewalls must function to block unauthorized data flows, i.e *Firewall.Functioning=True*. The attack step *Reconfigure* target a *Firewall* and will therefore disable them if accomplished. *Firewall* may refer to *AccessLogging* with the reference slot *Firewall.Has*. The class *AccessLogging* will influence the probability that *Reconfigure* leaves accountability. To specify this

Reconfigure.LeavesAccountability have parents *Reconfigure.Target.Has.Functioning*.

6.2 Malware and anti-malware

Services can be exploited to execute malware on the hosts that run them. *ExploitVulnerability* has the reference slot *TargetB*, where *Range(ExploitVulnerability.TargetB)=Service*, and *Host* have the reference slot *Host.Run* where *Range(Host.Run)=Service*. The reference slots *ExecuteMalware.Target* and *MalwareScanner.Monitor* have the range *Host*.

A service's vulnerability can only be exploited if a connection can be established to it. Hence, *ExecuteMalware.Target.Run.TargetB*$^{-1}$*.PossibleToAccomplish* is a parent of *ExecuteMalware.PossibleToAccomplish*. The attack step *ExecuteMalware* can be detected if the host is monitored with a *MalwareScanner*. If detected, *ExecuteMalware.PossibleToAccomplish* will be influenced by removal functionality on the host *RemovalFunctionality.Activated*. The rules for these dependency relationships are shown in Table 3.

Table 3. Parents to attributes in the Anti-malware part of the PRM.

ExploitVulnerability.PossibleToAccomplish	
	ExploitVulnerability.TargetB.TargetA^{-1}.PossibleToAccomplish
ExecuteMalware.PossibleToAccomplish	
	ExecuteMalware.Target.Run.TargetB^{-1}.PossibleToAccomplish
	ExecuteMalware.Target.Heal^{-1}.Activated
ExecuteMalware.Detected	
	ExecuteMalware.IsTargetIn^{-1}.Monitor^{-1}.Functioning
RemovalFunctionality.Activated	
	RemovalFunctionality.Functioning
	RemovalFunctionality.Heals.Target^{-1}.Detected

6.3 Social engineering and backup data

In addition to technical concepts, such as firewalls and network addresses, the AbstractPRM-package allows organizational and human elements of security to be included in ConcretePRM-packages. In this example (cf. Figure 7), the class *OrganizationalFunction* represents a role within the organization.

OrganizationalFunction has the reference slots *Use* and *CoveredBy*, where *Range(Use)=Host* and *Range(CoveredBy)=AwarenessProgram*.

An OrganizationalFunction can be targeted by the attack step MakeUserInstallExecutable, so Range(MakeUserInstallExecutable.Target)= OrganisationalFunction. The attribute ExecuteMalware.PossibleToAccomplish is dependent on whether it is possible to make the targeted host's users install an executable in a host. This in turn is influenced by whether this person is covered by a security awareness program.

Host has the reference slot *Contain* with range *DataStore*. The class *DataStore* has the reference slot *Has*, where *Range(DataStore.Has)=Backup*. The expected loss of a data store is influenced by if its backups are functioning. This value can for example represent the monetary loss associated with the conditions in parent-attributes. If general loss values can be identified, and these are expected to be as accurate as those that can be provided by the person applying the ConcretePRM-package, the loss values associated with successful attacks can be specified in the ConcretePRM-package. In this ConcretePRM-package the class *DataStore* is however vaguely defined. The monetary loss associated with the destruction of different *DataStore*-objects is this example collected together with the instance model instead. The question marks in the definition of this *DataStore.ExpectedLoss* mark that this information should be provided when the architecture model is instantiated.

Table 4 details the rules to determine parents of this attribute as well as other attributes in this part of the ConcretePRM-package. The parent *DataStore.Contain*$^{-1}$*.Target*$^{-1}$*.Includes*$^{-1}$*.IsRealized* refers to the *Threat* which causes the loss and is explained in more detail in section 6.4.

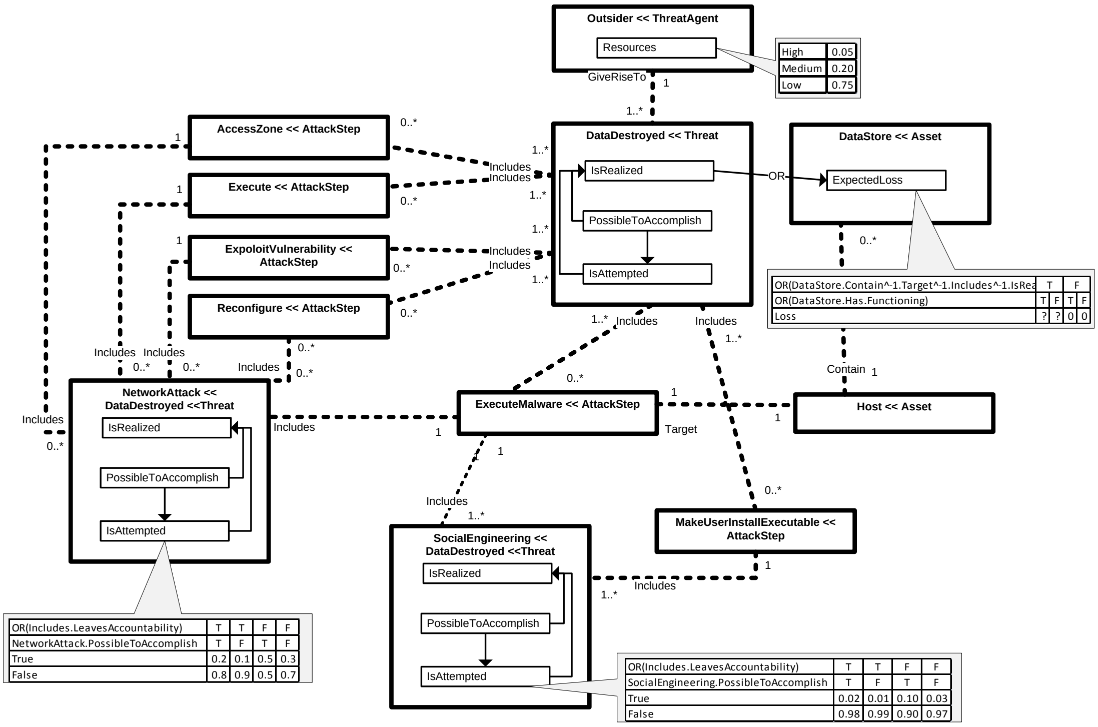
Table 4. Parents to attributes in the social engineering part of the PRM.

DataStore.ExpectedLoss
DataStore.Has.Functioning
DataStore.Contain^{-1}.Target^{-1}.Includes^{-1}.IsRealized
ExecuteMalware.PossibleToAccomplish
ExecuteMalware.IsTargetIn^{-1}.UsedBy.IsTargetIn.PossibleToAccomplish
MakeUserInstallExecutable.PossibleToAccomplish
MakeUserInstallExecutable.IsTargetIn^{-1}.CoveredBy.Functioning
DataStore.Contain^{-1}.Target^{-1}.Includes^{-1}.IsRealized

6.4 Threats and Threat Agent

The AbstractPRM-package allows a ConcretePRM-package to specialize *ThreatAgent* and *Threat* into subclasses. *ThreatAgent* can be specialized to define the probability distribution of the attribute *Resources*, and to define the semantics of its states. The class *Threat* can be specialized to: define *Threat.IsRealized* as a parent to some *Asset.ExpectedLoss*, to refine the reference slot *Threat.GiveRiseTo^{-1}*, to refine the reference slot *Threat.Includes*, and to specialize the conditional probability of *Threat.IsAttempted*.

Subclasses of *Threat* and *ThreatAgent* and how these relate to other classes in the ConcretePRM package is shown in Figure 8. The conditional probabilities for the *IsAttempted*-attributes here represent the probability that an attack will be attempted over the duration of one year.



High	0.05
Medium	0.20
Low	0.75

OR(DataStore.Contain^-1.Target^-1.Includes^-1.IsRea	T		F	
OR(DataStore.Has.Functioning)	T	F	T	F
Loss	?	?	0	0

OR(Includes.LeavesAccountability)	T	T	F	F
NetworkAttack.PossibleToAccomplish	T	F	T	F
True	0.2	0.1	0.5	0.3
False	0.8	0.9	0.5	0.7

OR(Includes.LeavesAccountability)	T	T	F	F
SocialEngineering.PossibleToAccomplish	T	F	T	F
True	0.02	0.01	0.10	0.03
False	0.98	0.99	0.90	0.97

Figure 8. Threats and ThreatAgent in the ConcretePRM-package. The refined range of the Includes-slot in SocialEngineering and NetworkAttack is here defined together with their multiplicities. The figure also shows specialized probability distributions and that DataDestoryed.IsRealized is a parent to DataStore.ExpectedLoss. Attributes of AttackStep-specializations and associated attribute dependencies is not shown in this figure.

DataDestoroyed is a subclass of *Threat*. The attribute *DataStore.ExpectedLoss* has parents *DataStore.Store*$^{-1}$*.Target*$^{-1}$*.Include*$^{-1}$*.IsRealized*, i.e. if this threat is realized *DataStore.ExpectedLoss* is influenced. We also let Outsider be a subclass of *ThreatAgent* and *Outsider.GiveRiseTo* be refined so that *Range(Outsider.GiveRiseTo)= DataDestroyed*. The probability distribution for *Outsider.Resources* is shown in Figure 8 and its states are defined as: (*High*) a professional cyber-criminal spending up to one week, (*Medium*) a professional cyber-criminal spending up to one hour, (*Low*) a beginner spending one hour.

To refine the *AttackSteps* a *Threat* might include we let *NetworkAttack* and *SocialEngineering* be two subclasses of

DataDestroyed, i.e. *NetworkAttack « DataDestroyed* and *SocialEngineering « DataDestroyed*. Figure 8 specifies the conditional probabilities for *IsAttempted* for these two threats. We let the range of *SocialEngineering.Includes* and *NetworkAttack.Includes* be defined as in Figure 8. These refinements do for instance say that *NetworkAttack.Includes* may refer to any number of *Reconfigure*-instances, but not to an instance of *MakeUserInstallExecutable*.

7 Instance model and security risk analysis

The AbstractPRM-package provides a basis for creating metamodels that supports probabilistic inference of security risk. This chapter exemplifies how inference is supported by applying the ConcretePRM-package described in chapter 6 to a case study.

The case study and the resulting instance model are described in section 7.1. In section 7.2 it is shown how probabilistic reachability analysis can be performed on the instance model. This method of analysis is similar to the reachability analysis performed on attack graphs and produce probabilities on the possibility to accomplish different attack steps. The AbstractPRM-package does in addition to this also allow inference of expected loss. Section 7.3 describe how expected loss can be inferred from instance models that instantiate ConcretePRM-packages.

7.1 Case study

The case study was carried out at an electric power utility in Sweden during November 2009. The scope of this study was one of the critical substations within the utility. This substation was modeled with ConcretePRM-package described in chapter 6. To create the instance model, interviews with system administrators were conducted and system documentation was investigated.

To create the instance model subclasses of *Asset* and reference slots where both range and target is an *Assets* (i.e. refinements of *Asset.Association*) must be specified. This, together with loss values of different *DataStore*-objects, was the only information collected in the case study. In Figure 9 white boxes represent the objects that instantiate a subclass of *Asset*. The dashed lines between these white boxes are the instantiated reference slots.

Four relevant *Zones* are located outside of the substation. The *OfficeNetwork* is the insecure side of the *CorporateFirewall*. The *CorporateFirewall* allow the data flow *RemoteDesktop* to pass through from the *OfficeNetwork* to the service *TerminalServices*, and the data flow *SubstationCommunication* from the zone *ControlCenterNetwork* to reach *ControlSystemServer*. The *GatewayFirewall* is connected to the Internet and allow data to pass from both the *OfficeNetwork* and the *SuppliersLAN*.

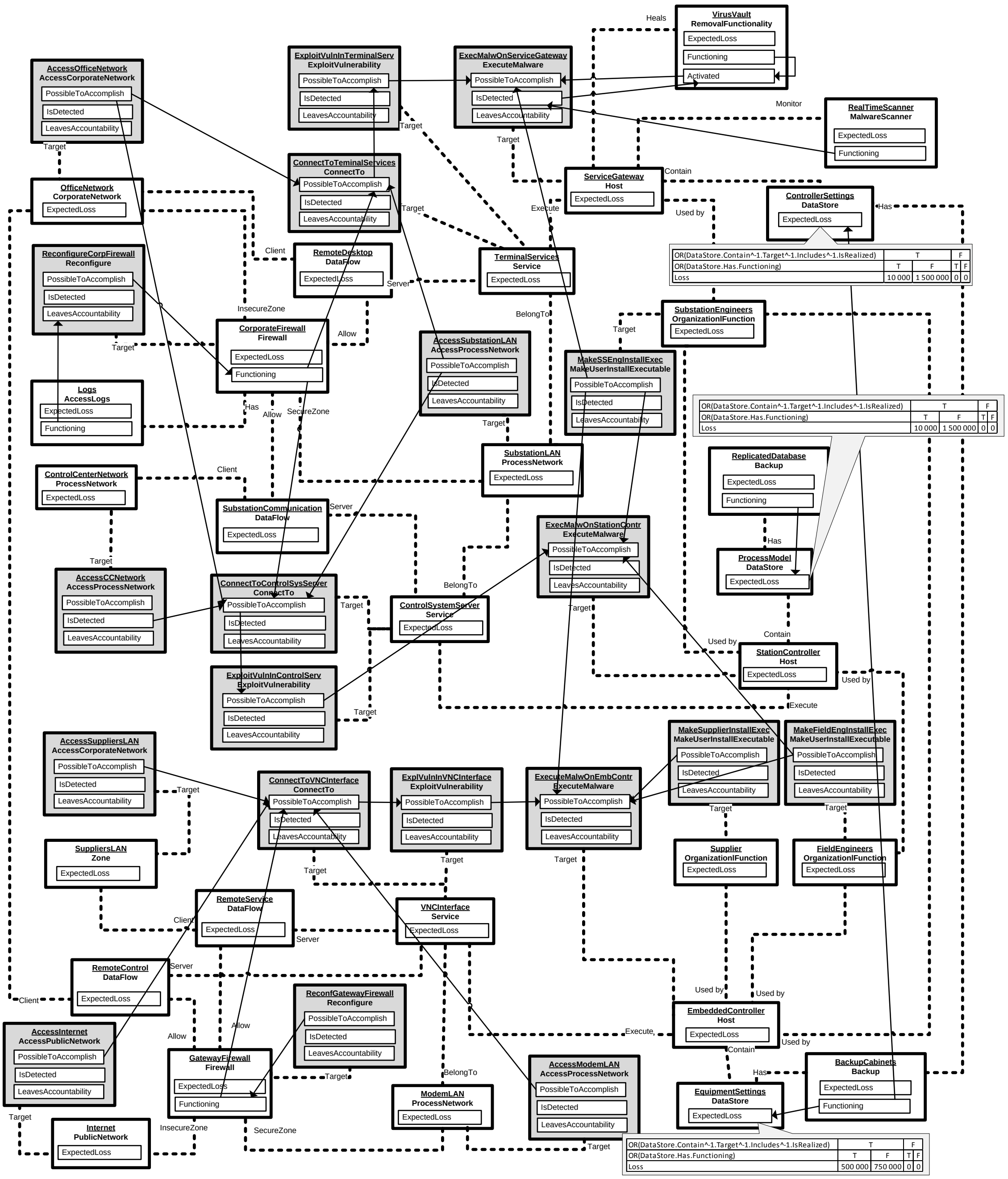
Within the substation there are two process networks: the *SubstationLAN* and the *ModemLAN*. The services *ControlSystemServer* and *TerminalServices* belong to the *SubstationLAN*. The *ControlSystemServer* is executed by the host *StationController* and is the server for *SubstationCommunication*. The *StationController* contain a *ProcessModel*. This data store has a backup – the *ReplicatedDatabase*. The service *TerminalServices* is executed by the host *ServiceGateway* and is the server for the data flow *RemoteDesktop*. The *ServiceGateway* contains *ControllerSettings* and is used by *SubstationEngineers* to reconfigure information technology within the substation. The *ServiceGateway* is monitored by a *RealTimeScanner* and a *VirusVault* can heal it if malware is found on it. The service *VNCInterface* belongs to the ModemLAN and is executed by the host *EmbeddedController*. This *EmbeddedController* contains another data store – *EquipmentSettings*. In a *BackupCabinet* there are backups of both *EquipmentSettings* and *ControllerSettings*.

Three organizational functions use hosts within the substation: *Supplier*, *FieldEngineers* and *SubstationEngineers*. The *EmbeddedController* is used by Supplier and *FieldEngineers*; the *ServiceGateway* is used by *SubstationEngineers*; the *StationController* is used by *SubstationEngineers* and *FieldEngineers*. None of these are covered by an awareness program.

The tables in Figure 9 present the loss (in Swedish krona, SEK) that would be experienced if the data stores would be destroyed. These state the loss under the conditions that a backup exist, and that it does not exist. As can be seen from Figure 9 the impact of backups on the expected loss is substantial for the attributes *ControllerSettings.ExpectedLoss* and *ProcessModel.ExpectedLoss*. The impact from a backup on the attribute *EquipmentSettings.ExcepectLoss* is less, both in relative and absolute terms. This is due to the complicated nature of these settings and the manual labor required to restore them from a backup.

The *AttackSteps* relevant for this architecture model can be derived and instantiated based on the *Asset*-objects and the reference slots between *Asset*-objects. The multiplicities associated with reference slots constrain the instance model, and provide support for deriving the attack steps that should be included in an instantiation. As suggested by the AbstractPRM, the *AttackStep*-subclasses in the ConcretePRM-package refer to exactly one *Asset* with the reference slot *Target*. This makes it possible to identify and instantiate the *AttackStep*-objects that an instantiated *Asset*-subclass should be associated to – for each *Asset*-object in the instance model, an object of each class in *Range(Asset.Target^{-1})* should be added and referred. Hence, based on instantiations of *Assets* and asset-references, associated *AttackStep* instances can be derived. The attack steps associated with this architecture models are colored grey in Figure 9.

With this instance model as a basis, different architectural changes can be assessed in terms of their impact on reachability and expected loss. The ConcretePRM-package used in this case study specifies that only organizational functions that use a host can be made to install executables on them. It also specifies that if it is more difficult to make users install executables if their organizational functions are covered by awareness programs. Two architectural changes were assessed in this case study: 1) the impact of making sure that substation engineers do not use the embedded controller and 2) covering substation engineers with an awareness program. Objects instantiating *Owner*, *Threat* and *ThreatAgent* are not shown in Figure 9. These are instead introduced in section 7.2 and 7.3, together with the analysis.



OR(DataStore.Contain^-1.Target^-1.Includes^-1.IsRealized)	T		F	
OR(DataStore.Has.Functioning)	T	F	T	F
Loss	10 000	1 500 000	0	0

OR(DataStore.Contain^-1.Target^-1.Includes^-1.IsRealized)	T		F	
OR(DataStore.Has.Functioning)	T	F	T	F
Loss	10 000	1 500 000	0	0

OR(DataStore.Contain^-1.Target^-1.Includes^-1.IsRealized)	T		F	
OR(DataStore.Has.Functioning)	T	F	T	F
Loss	500 000	750 000	0	0

Figure 9. The case study represented through the ConcretePRM depicted in Figure 7. AttackStep-instances (toned gray) can be derived from the ConcretePRM-package together with their references.

7.2 Probabilistic reachability analysis

The dependencies between objects of class *Countermeasure* and *AttackStep* is given by an instance model. These relationships make it possible to create a probabilistic graph for reachability analysis where the possibility. This reachability analysis assesses the possibility to accomplish different attack steps targeting assets in the instance model.

If *I* is the architecture instantiation of a ConcretePRM, let I^{RA} include all objects in *I* that are of class: *AttackStep*, *PreventiveCounteremeasure*, *DetectiveCountermeasure* or *ReactiveCountermeasure*, including their attributes and internal attribute dependencies. Also let the objects *T* and *TA* be included in I^{RA}, where *T* is an instance of a subclass to *Threat* (T ∈ *C[Threat]*) and *TA* be an instance of a subclass to *ThreatAgent* (*TA* ∈ *C[ThreatAgent]*). Let *T.Includes* point to all *AttackStep*-objects in its range, and let *TA.GiveRiseTo* point to *T*.

This yields a graph, I^{RA}, which can be used to infer the probabilities of attributes *PossibleToAccomplish* and *Detected* in all instances of *AttackStep*. This inference is performed under the assumption that all attack steps are attempted.

Figure 10 depicts the graph I^{RA}, excluding *TA* and *T*, for the instance model *I* in Figure 9. Let *T* be of class *DataDestroyed,* and *TA* be of class *Outsider*. An instance of *DataDestroyed* (*InstanceOfTA*) then refers to all *AttackStep*-objects in Figure 9 with the reference slot *Includes*, and an instance of *Outsider* (*InstanceOfT*) refers to the instance of *DataDestroyed.*

The probabilistic model associated with I^{RA} can for example be used to infer if attack steps are possible to accomplish given different *TA.Resources*, or to infer which attack steps that can be accomplished when the attributes of attack steps and countermeasures are set to specific values, i.e. evidence has been provided on the state of these attributes. In Figure 10 the reachability analysis has been performed under the assumption that threat agent has medium resources, i.e. *TA.Resources=Medium*. Figure 10 does for instance show that *P(ExecuteMalwOnEmbContr.PossibleToAccomplish)=50 %* under these conditions. It also shows that the possibility to exploit vulnerabilities in the services is significantly smaller than the possibility to make users install executables on hosts.

Two architectural modifications were assessed in this case study. The first architectural modification was to make sure that substation engineers do not use the embedded controller.

This would according to the ConcretePRM-package reduce *P(ExecuteMalwOnEmbContr.PossibleToAccomplish)* from 50 % to 37 %. The second modification, covering substation engineers in an awareness program, would reduce *P(MakeSSEngInstallExec.PossibleToAccomplish)* from 20 % to 10%. This would in turn reduce *P(ExecuteMalwOnServiceGateway.PossibleToAccomplish)* to 10 %, *P(ExecMalwOnStationContr.PossibleToAccomplish)* to 28 % and *P(ExecuteMalwOnEmbContr.PossibleToAccomplish)* to 44 % . The impact of architectural changes like these on reachability is assessed by adding or removing assets and asset-to asset relationships in the instance model.

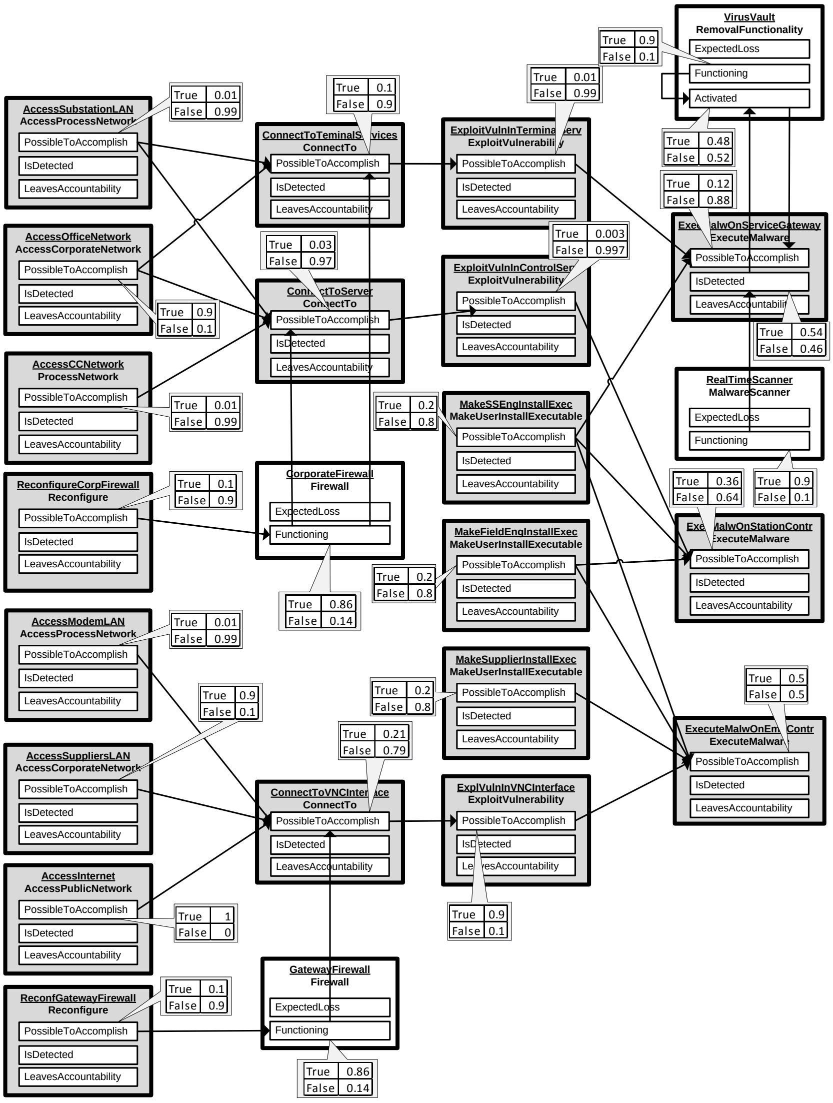

VirusVault RemovalFunctionality
AccessSubstationLAN AccessProcessNetwork
ConnectToTeminalServices ConnectTo
ExploitVulnInTerminalServ ExploitVulnerability
ExecMalwOnServiceGateway ExecuteMalware
AccessOfficeNetwork AccessCorporateNetwork
ConnectToServer ConnectTo
ExploitVulnInControlServ ExploitVulnerability
AccessCCNetwork ProcessNetwork
MakeSSEngInstallExec MakeUserInstallExecutable
RealTimeScanner MalwareScanner
ReconfigureCorpFirewall Reconfigure
CorporateFirewall Firewall
MakeFieldEngInstallExec MakeUserInstallExecutable
ExecMalwOnStationContr ExecuteMalware
AccessModemLAN AccessProcessNetwork
MakeSupplierInstallExec MakeUserInstallExecutable
AccessSuppliersLAN AccessCorporateNetwork
ConnectToVNCInterface ConnectTo
ExplVulnInVNCInterface ExploitVulnerability
ExecuteMalwOnEmContr ExecuteMalware
AccessInternet AccessPublicNetwork
GatewayFirewall Firewall
ReconfGatewayFirewall Reconfigure
PossibleToAccomplish
IsDetected
LeavesAccountability
ExpectedLoss
Functioning
Activated

Figure 10. Reachability graph comprising of attack steps, preventive countermeasures, detective countermeasures and reactive countermeasures along with their attribute-relationships. Here Adversary.Resources (not shown in the figure) is set to medium.

7.3 Loss expectancy

The method described in section 7.2 allows inference of the probability that attack steps are possible to accomplish under the assumption that the threat agent attempts to perform all attack steps. Although the probabilities that are inferred with this analysis method provide an indication of security, just as reachability analysis performed on attack graphs and attack trees, it does not capture security risk (*Owner.ExpectedLoss*). If risk is the variable sought, it should also be assessed if the threats will be attempted (*Threat.IsAttempted*) and threats should be related to the losses they cause when realized. Countermeasures that deter threat agents from attempting attacks (*AccountabilityCountermeasure*) and countermeasures that limit the loss (*ContingencyCountermeasure*) are also of relevance when this is assessed. By instantiating the threats one wish to include in the analysis these factors can be included in the analysis. The expected loss from different threats can be assessed. In this case study we instantiated objects the classes of *NetworkAttack* and *SocialEngineering*.

In Figure 11 two *Threat*-instances are shown. In the first *Threat1* includes *MakeSSEngInstallExec* and *ExecMalwOnServiceGateway*; in the second *Threat2* includes *MakeSSEngInstallExec* and *ExecMalwOnStationContr*. For each plausible instance like these, the probability *P(SocialEngineering.IsRealized)* can be inferred and the value for the expected losses associated with different assets can be calculated. The probabilistic dependency structure infers the expected losses 619 SEK and 1385 SEK for these two threats.

In this case study 17 different threats were found relevant for the substation in question. Six are of the class *SocialEngineering* and eleven of the class *NetworkAttack*. Table 5 and Table 6 list these together with the expected losses associated with them. Creating the object *ElectricUtility* of class *Owner* and referring this to all *DataStore*-instances aggregates these losses to the attribute *ElectricUtility.ExpectedLoss*.

If the architecture is changed so that substation engineers do not use the embedded controller this influences the third threat (T3). T3 would in this case be associated with an expected loss of zero. Covering substation engineers in an awareness program would influence threat one (T1), two (T2) and three (T3). The expected losses from these threats would be reduced to 390 SEK, 689 SEK and 2275 SEK respectively. With these predictions as a basis, the first architectural change lowers expected losses by 4573 SEK and the second alternative would reduce losses by SEK 3224.

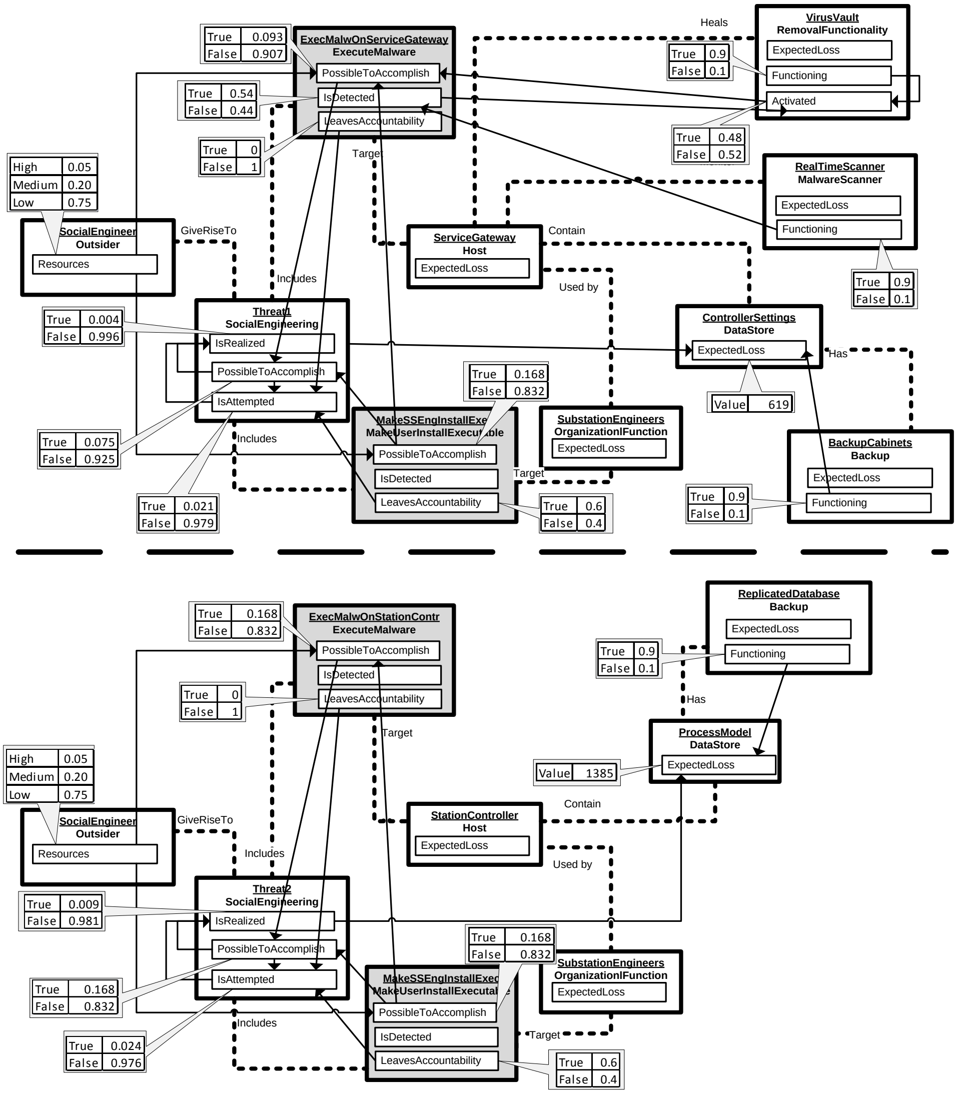


Figure 11. Example of instantiated SocialEngineering-threats that corresponds to Threat1 and Threat2. Only those objects that are of relevance to these threats are included in the figure.

Table 5. Instantiations of SocialEngineering and associated expected loss. An X denotes that the attack step is included in the threat.

AttackStep\Threat	T1	T2	T3	T4	T5	T6
MakeSSEngInstallExec	●	●	●			
MakeSupplierInstallExec				●		
MakeFieldEngInstallExec					●	●
ExecMalwOnServiceGateway	●					
ExecuteMalwOnStationContr		●			●	
ExecuteMalwareOnEmbContr			●	●		●
Expected loss ($)	**620**	**1385**	**4573**	**689**	**1385**	**4573**

Table 6. Possible references and assessed instantiations of NetworkAttack. An X denotes that the attack step is included in the threat.

AttackStep\Threat	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
AccessOfficeNetwork	●	●		●	●						
AccessCCNetwork						●	●				
AccessSubstationLAN			●					●			
AccessSuppliersLAN									●		
AccessInternet										●	●
AccessModemLAN											
ReconfigureCorpFirewall		●			●		●				
ReconfGatewayFirewall											●
ConnectToTerminalServices	●	●	●								
ConnectToControlSystemServer				●	●	●	●	●			
ConnectToVNCInterface									●	●	●
ExploitVulnInTerminalServ	●	●	●								
ExploitVulnInControlServ				●	●	●	●	●			
ExploitVulnVNCInterface									●	●	●
ExecMalwOnServiceGateway	●	●	●								
ExecuteMalwOnStationContr				●	●	●	●	●			
ExecuteMalwareOnEmbContr									●	●	●
Expected loss ($)	**379**	**243**	**57**	**71**	**75**	**94**	**60**	**790**	**1851**	**279**	**810**

8 Discussion

This paper proposes the use of PRMs to perform security risk analysis based on architecture models and present a package of abstract base classes for PRMs with this purpose. This chapter will discuss the expressiveness of the proposed class-structure and some practical issues associated with the suggested approach.

The AbstractPRM-package presented herein prescribes the classes, attributes, class-references, and attribute dependencies a ConcretePRM-package should include. Although the AbstractPRM-package thereby constrains the structure of a ConcretePRM-package, it does not dictate what level of detail a ConcretePRM-package should have. A ConcretePRM-package could include attack steps on abstraction levels such as “Run code”, “Exploit buffer overflow vulnerability to run code” or “Exploit VU#238019 to run arbitrary code”. Similarly, a ConcretePRM-package can specify assets, countermeasures, threats and threat agents on any level of abstraction. This flexibility is a result of the possibility to represent uncertain variable-relationships probabilistically. A more detailed model would typically enable more informative conditional probability distributions. Given that the conditional probabilities are accurate this would make a ConcretePRM-package able to produce more informative predictions when it is instantiated. For example, a detailed ConcretePRM-package might be able to predict the possibility to accomplish with an attack to 96% or 1 % for two architectures. A less detailed ConcretePRM-package might be able to predict this probability to 70% or 35 %.

Although detailed ConcretePRM-package would allow more predictive power, more effort is required both when the ConcretePRM-package is created and when the Concrete PRM-package is instantiated. Hence, informative predictions must be balanced against cost of identifying accurate conditional probabilities for it and the practitioners’ cost when creating instance models with it. Another trade-off is the utility and relevance of the PRM’s predictions to practitioners. A PRM that only covers a limited scope might be able to offer more informative predictions than one with a wider scope and less detail. However, a limited scope increases the risk of sub optimizing the architecture design decisions from an enterprise-wide perspective.

One feature of the proposed AbstractPRM-package is that it includes inference of the probability that attacks are attempted. The threat agent’s decision model is here compactly represented by a probability distribution which states if the threat will be attempted given the probability that the attack succeeds and that it leads to accountability. Properties of threat agents are also compactly represented in the AbstractPRM. Clearly, the representation of threat agents' resources on the scale High/Medium/Low cannot capture all distinguishing characteristics of threat agents. This attribute is however, just as threat agents' decision model, a research field on its own. This AbstractPRM-package does not elaborate on these two fields, but instead provides a clear-cut interface to them.

A software tool [49] has been developed to support the creation and instantiation of PRMs based on the inference engine SMILE [53]. Instantiated PRMs can become quite large, but there exist today methods for solving also very large Bayesian networks [54]. Future extensions of this tool include support to automatically instantiate relevant attack steps and threats for a particular instance model. To be of practical use for decision makers though, ConcretePRM-packages must be specified. The dependencies among variables in the security field can be obtained from sources such domain experts, literature, experiments, vulnerability statistics, security exercises (e.g. red team-blue team exercises), or a combination of sources like these. Models can also be updated as new threats or countermeasures emerge.

To create a ConcretePRM-package where the qualitative structure is optimal with regard to security risk prediction and the conditional probabilities represent an undisputable truth is of course extremely difficult, if not impossible. If domain experts judgment has been used to define conditional probabilities its output will also be of subjective nature. However, to be of practical use it is sufficient if a ConcretePRM-package captures the knowledge that is available in the security field and thereby provides the decision maker with a tool that improves security risk analysis activities. Also, decision makers are often interested in how different architectural scenarios are ordered when it comes to security, for example if the to-be architecture is better or worse than the as-is architecture. In that case the exact values of the predictions are not their most important quality, but rather that scenarios are correctly ordered with respect to their security risk.

The mere possibility to express and quantify how security theories relate to different architectures is another feature offered by the AbstractPRM-package. Security theories from diverse fields can be expressed in ConcretePRM-packages, and these theories can be consolidated with each other by integrating their packages. Furthermore, since PRMs in their pure form are versatile, ConcretePRM-packages can also be integrated with PRMs that express theories from other fields. For example theories on how costs, business value or modifiability relates to different architectures. This would allow decision makers to make informed decisions regarding the security risk associated with different enterprise architectures; while at the same time take other concerns into consideration.

9 Conclusions

PRMs allow architectural metamodels to be coupled to a probabilistic inference engine. This makes it possible to specify how the state of object's attributes depends on the state of other attributes in an architectural model. This paper proposes a package of abstract PRM-classes that specify how probabilistic models should be coupled to architectural metamodels to enable security risk analysis.

The versatility provided by the probabilistic side of PRMs makes it possible to specify security theories on any abstraction level and to couple these with an architectural metamodel with the proposed set of abstract classes. Concrete classes can be developed by creating subclasses to the package of abstract classes and under a set of constraints associate these with a probabilistic model. By specializing the abstract classes into concrete classes an architectural metamodel is defined and associated with formal machinery for assessing security risk from its instantiations. The structure inherited from the abstract classes also ensures that this formal machinery can calculate security risk from an instance model that only specifies assets and asset-relationships specified in an architectural model. Hence, the person instantiating the instance model is not required to quantify security attributes or provide information on vulnerabilities for security risk to be assessed.

10 References

[1] J. J. C. H. Ryan, D. J. Ryan, Expected benefits of information security investments, Computers & Security, 25 (2006) 579-588.

[2] T. Tsiakis, G. Stephanides, The economic approach of information security, Computers & Security, 24 (2005) 105-108.

[3] H. Cavusoglu, B. Mishra, and S. Raghunathan, A model for evaluating it security investments, Communications of the ACM, 47 (2004) 87–92.

[4] L. A. Gordon, M. P. Loeb, Managing Cybersecurity Resources: A Cost-Benefit Analysis, Mcgraw-Hill, New York, USA, 2006.

[5] W. Ozier, Risk analysis and assessment, in Information security management handbook, 4th ed, Auerbach, Boca Raton, USA, 1999, pp. 247–285.

[6] W. Huaqiang, F. Deb, C. Olivia, R. Chris, Cost benefit analysis for network intrusion detection systems, CSI 28th annual computer security conference, Washington, DC, 2001.

[7] C. Iheagwara, The effect of intrusion detection management methods on the return on investment, Computers and Security, 23(2004) 213–228.

[8] Hogganvik, A Graphical Approach to Security Risk Analysis. Oslo, Norway, Norway: University of Oslo - Faculty of Mathematics and Natural Sciences, 2007.

[9] O. Sheyner, Scenario Graphs and Attack Graphs. PhD Thesis, Carnegie Mellon University, 2004.

[10] N. Friedman, L. Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in In International Joint Conferences on Artificial Intelligence, pp. 1300-1309, 1999.

[11] Object Management Group (OMG), Unified Modeling Language (UML), 2009.

[12] Object Management Group (OMG), OMG Systems Modeling Language (OMG SysML), 2008.

[13] Object Management Group (OMG), Business Process Modeling Notation, 2009.

[14] G. Sindre, A. L. Opdahl, Eliciting security requirements with misuse cases, Requirements Engineering, 10 (2005) 34–44.

[15] J. McDermott and C. Fox, Using abuse case models for security requirements analysis, in Proceedings of the 15th annual computer security applications conference, pp. 55, Phoenix, Arizona, USA, 1999.

[16] J. McDermot, Abuse-case-based assurance arguments, in Proceedings of the 17th annual computer security applications conference, pp.0366, New Orleans, Los Angeles, USA, 2001.

[17] T. Lodderstedt, D. Basin, J. Doser, SecureUML: A UML-Based Modeling Language for Model-Driven Security, in «UML» 2002 - The Unified Modeling Language. Springer Berlin / Heidelberg, 2002.

[18] K. M. Trevisani, R. E. Garcia, SPML: A Visual Approach for Modeling Firewall Configurations, in Modeling Security Workshop, Toulouse, France, 2008. Avaliable at: http://www.comp.lancs.ac.uk/modsec/papers/modsec08_submission_21.pdf

[19] H. Mouratidis, P. Giorgini, G. Manson, I. Philp, A Natural Extension of Tropos Methodology for Modelling Security, in Proceedings of the Agent Oriented Methodologies Workshop, Seattle, USA, 2002.

[20] J. Jürjens, Secure Systems Development with UML. Berlin Heidelberg: Springer-Verlag, 2005.

[21] ISO/IEC JTC1/SC27, Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and general model, CCMB 2006-09-01, 2006.

[22] Insight Consulting, The Logic behind CRAMM'sAssessment of Measures of Risk and Determination of Appropriate Countermeasures, 2005.

[23] B. Karabacak, I. Sogukpinar, ISRAM: information security risk analysis method, Computers & Security, 24 (2005), 147-159.

[24] C. J. Alberts, A. J. Dorofee, OCTAVE Criteria Version 2.0, CMU/SEI-2001-TR-016. ESC-TR-2001-016, 2001.

[25] M. Howard, D. C. LeBlanc, Writing Secure Code, Microsoft Press, Redmond, WA, USA, 2002.

[26] S. E. Schechter, Computer Security Strength & Risk: A Quantitative Approach, PhD Thesis, Harvard University, Boston, USA, 2004.

[27] B. Schneier., Attack trees: Modeling security threats, Dr. Dobb's Journal , December, 1999.

[28] S. Jha, O. Sheyner, J. Wing, Two formal analyses of attack graphs, in Proceedings of the 15th Computer Security Foundation Workshop, pp. 49-63, 2002.

[29] P. Pamula, P. Ammann, A.Jajodia, V. Swarup, A weakest-adversary security metric for network configuration security analysis, in Conference on Computer and Communications Security, Proceedings of the 2nd ACM workshop on Quality of protection, pp. 31 - 38 ,2006.

[30] P. Ammann, D. Wijesekera, S. Kaushik, Scalable, graph-based network vulnerability analysis, in Proceedings of 9th ACM Conference on Computer and Communications Security, pp. 217 - 224, 2002.

[31] S. Jajodia, S. Noel, B. O'Berry, Topological analysis of network attack vulnerability, in Managing Cyber Threats: Issues, Approaches and Challanges, chapter 5. Kluwer

Academic Publisher, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. Springer US, 2003, pp. 247-266.

[32] T. Tidwel, R. Larson, K. Fitch, J. Hale, Modeling Internet attacks, in IEEE Workshop on Information Assurance and Security, pp 54-59, West Point, NY, USA, 2001.

[33] C. Phillips, L. P. Swiler, A graph-based system for network-vulnerability analysis, in Proceedings of the 1998 workshop on New security paradigms, pp. 17-79, 1998.

[34] X. Ou, W. F. Boyer, M. A. McQueen., A Scalable Approach to Attack Graph Generation, in Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 336-345, 2006.

[35] Y. Liu, M. Hong, Network vulnerability assessment using Bayesian networks, in proceedings of SPIE, , pp. 61-71Orlando, Florida, USA, 2005.

[36] W. J. Caelli, D. Longley, A. B. Tickle, A methodology for describing information and physical security architectures, in Proceedings of the IFIP TC11, Eigth International Conference on Information Security, vol. A-15, pp. 277–296, Singapore, 1992.

[37] Anderson, D. Longley, L. F. Kwok, Security modelling for organisations, in Proceedings of the 2nd ACM Conference on Computer and communications security, pp. 241-250, Fairfax, Virginia, United States, 1994.

[38] S. Bistarelli, F. Fioravanti., P. Peretti., Defense trees for economic evaluation of security investments, in in proceedings of The International Conference on Availability, Reliability and Security, pp. 416-423, Vienna, Austria, 2006.

[39] S. Bistarelli, M. Dall'Aglio, P. Peretti, Strategic games on defense trees, in Formal Aspects in Security and Trust. Springer Berlin / Heidelberg, 2007, pp. 1-15.

[40] S. Bistarelli, F. Fioravanti, P. Peretti, Using CP-nets as a Guide for Countermeasure Selection, in Proceedings of the ACM symposium on Applied computing, pp. 300 – 304, Seoul, Korea, 2007.

[41] O. Sheyner, J. Wing, Tools for Generating and Analyzing Attack Graphs, in Formal Methods for Components and Objects. Springer Berlin / Heidelberg, 2004, pp. 344-371.

[42] L. P. Swiler, C. Phillips, D. Ellis, S. Chakerian, Computer-attack graph generation tool, DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings , vol.2, pp.307-321, 2001.

[43] R. W. Ritchey, P. Ammann, Using model checking to analyze network vulnerabilities, in Proceedings of the IEEE Symposium on Security and Privacy, pp. 156–165, 2001.

[44] T. Sommestad, M. Ekstedt, P. Johnson, Cyber Security Risks Assessment with Bayesian Defense Graphs and Architectural Models, in 42nd Hawaii International Conference on System Sciences, pp. 1-10, Hawaii, 2009.

[45] F. V. Jensen, An Introduction to Bayesian Networks, New York: Springer-Verlag, 1996.

[46] W. H. Hsu, R. Joehanes, Relational Decision Networks, in Working Notes of the ICML-2004 Workshop on Statistical Relational Learning and Connections to Other Fields, pp. 61-67, Banff, Canada, 2004.

[47] R. Shachter, Evaluating influence diagrams, Operations Research, 34 (1986) 871-882.

[48] W. Schwartau, Time-based security explained: Provable security models and formulas for the practitioner and vendor, Computers & Security, 17 (1998), 693-714.

[49] M. Ekstedt, U. Franke, P. Johnson, R. Lagerström, T. Sommestad, J. Ullberg, M. Buschle, A Tool for Enterprise Architecture Analysis of Maintainability. In Proceedings of the 2009 European Conference on Software Maintenance and Reengineering, 327-328,Washington, DC, USA, 2009,

[50] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modeling security requirements through ownership, permission and delegation, in 13th IEEE International Conference on Requirements Engineering, 167- 176, Paris, France, 2005.

[51] J. Jürjens, P. Shabalin, Automated Verification of UMLsec Models for Security Requirements, 2004 - The Unified Modelling Language, pp. 365-379, Springer Berlin/Heidelberg, 2004.

[52] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An Agent-Oriented Software Development, Autonomous Agents and Multi-Agent Systems , 8(2004), 203-236

[53] SMILE. Decision System Laboratory, University of Pittsburgh, http://genie.sis.pitt.edu/

[54] C.Yuan M.J Druzdzel. Mathematical and Computer Modelling, 43(2006), 1189-1207.

Paper B: Effort estimates for vulnerability discovery projects

Teodor Sommestad, Hannes Holm and Mathias Ekstedt

Abstract

Security vulnerabilities continue to be an issue in the software field and new severe vulnerabilities are discovered in software products each month. This paper analyzes estimates from domain experts on the amount of effort required for a penetration tester to find a zero-day vulnerability in a software product. Estimates are developed using Cooke's classical method for 16 types of vulnerability discovery projects – each corresponding to a configuration of four security measures. The estimates indicate that, regardless of project type, two weeks of testing are enough to discover a software vulnerability of high severity with fifty percent chance. In some project types an eight-to-five-week is enough to find a zero-day vulnerability with 95 percent probability. While all studied measures increase the effort required for the penetration tester none of them have a striking impact on the effort required to find a vulnerability.

1 Introduction

A substantial share of the security problems encountered in enterprises today arises because software products have security vulnerabilities. New vulnerabilities are discovered on a continuous basis. During 2010 alone, a total of 2096 new software vulnerabilities of high severity were publicly announced [1]. Many factors influence the number of vulnerabilities that are found in a software product. The effort invested into searching for vulnerabilities in a software product is one important variable [2], [3]. Another important variable is the difficulty associated with finding vulnerabilities in the software product, i.e. how much effort that is required to find a vulnerability in it.

Secure software development practice (see [4] for an overview) suggests a wide range of measures to increase the security of a software product's source code and thus increase the effort required to find a vulnerability, e.g. testing during the development phase. A natural question to ask is how much effort that is required to find a vulnerability in a software product given that different security enhancing measures have been used. Unfortunately, there are no studies available which answer this question, or even provide rough estimates of it. Ideally, this would be tested in experiments or derived from representative archival data on projects that attempted to discover vulnerabilities. However, constructing experiments of this kind are associated with substantial cost, and reliable archival data on efforts made not available to the community [5].

Expert judgment is often used when quantitative data is difficult to obtain from experiments or studies of archival data. This paper presents expert estimates on vulnerability discovery effort that are constructed using Cooke's classical method. This method assigns weights to experts based on how correct and certain they are on a set of questions related to the issue investigated, and for which the true answer is known at the time of analysis. It has been used to assess uncertain quantities in a wide range of domains and in general outperforms other methods that synthesize or aggregate domain experts' judgment [6].

The effort estimates presented in this paper quantify the effort associated with finding a zero-day vulnerability in a software product. That is, finding a vulnerability in deployed software product which is not already publicly announced or patched [7]. The experts in this study are researchers in the software vulnerability field. They used their domain knowledge to assess the work effort it takes for a professional penetration tester taking on 16 hypothetical vulnerability discovery projects, all with the goal to find a zero-day vulnerability of high severity. The resulting estimates show the probability that a vulnerability is found as a function of the work days spent on the project.

The paper is structured as follows. Section 2 presents the variables used in the effort estimation model. In section 3 Cooke's classical method is explained. Section 4 presents the method and section 5 presents the results. In section 6 these results are discussed and in section 7 conclusions are drawn.

2 Model and assumptions

This paper estimates the effort that is required to discover a zero-day vulnerability in a software product given that different security measures are used. Both the software security field and effort estimation field are well explored. However, no previous work has been found on the work-effort required to find zero-day vulnerabilities. This section presents the variables assessed in this study and the assumptions it is based upon.

2.1 Variables impacting discovery effort

A countless number of variables can be assumed to influence the effort required to find vulnerabilities in it. Technical measures, process measures and organizational measures are all of relevance [4].

Naturally, the scope of this research does not include all variables that could have an impact on the effort required to find a zero-day vulnerability. To identify a manageable set of variables to include a panel consisting of three security experts were consulted. All experts in this panel had practical experience of penetration testing and worked with security testing on a regular basis. They prioritized a list of candidate variables drawn from literature such as [4], [8–11]. They were also given the option to suggest variables not included in the list presented to them. Table 1 shows the variables that came out of this process and are included in this study. All these variables were expected to have an impact on the effort required to find a new vulnerability in a software product.

Table 1. Variables studied.

Variable	Description
Scrutinized	The targeted software has been scrutinized before.
SourceCode	The professional penetration tester has access to the source code.
SafeLanguage	The software is written in a safe language (e.g. C#, Java) or a safe dialect (e.g. Cyclone).
CodeAnalyzers	The software has been analyzed by static code analyzers and improved based on the result.

All variables described in Table 1 have support in literature. Software which has been *scrutinized* and tested in practice will be more difficult to find vulnerabilities in. This type of effect is often assumed in software reliability models [5] and data on vulnerabilities found in software products imply that a saturation level for vulnerabilities discovered in a product is reached after a certain time on the market [12]. Access to the *source code*, i.e. the uncompiled code, is also considered a relevant factor [13]. Access to the uncompiled source code will enable white box testing and is likely to decrease the effort required to find a vulnerability. If the programming language used to create the software product is a *safe language* [14] many potential programming flaws leading to vulnerabilities can be avoided. Finally, the use of *code analyzers* is often a recommended practice in software development to identify vulnerabilities in the code [15–17] .

2.2 Assumptions

A number of assumptions are used for the effort estimates produced in this study and are kept constant in this study. First, the competence of the actual performer of the vulnerability discovery project can be expected to have a substantial impact on the effort required [5]. To eliminate variations caused by this variable it is assumed that the person who carries out the vulnerability discovery project is a professional penetration tester. Secondly, it would be extremely difficult to find vulnerabilities in a product which is completely inaccessible to the person carrying out the project. Therefore, it is assumed that the person searching for vulnerabilities has access to the compiled code (the binary) even if the source code (*SourceCode*) is unavailable to him/her. Third, a work day was set to eight hours of work. This was specified to avoid confusion about what quantity that should be estimated (calendar, budget or effort) [18]. Fourth, the vulnerability that should be discovered needs to qualify as a high severity-vulnerability according to the Common Vulnerability Scoring System [19]. Since such vulnerabilities are more severe than other vulnerabilities it is more interesting to obtain knowledge about them. The final assumption used, and presented to those who estimated effort, was that all unspecified variables (e.g. the size of the source code) assume the state they typically have in an enterprise environment. Thus, any uncertainty remaining after the variables and assumptions are specified should be accounted for in the estimates. That is, variation between software not covered by the assumptions or variables will introduce uncertainty and variation to the effort required. The respondents were asked to consider this uncertainty onto the estimates the made.

3 Synthesizing expert judgments

There is much research on how to combine, or synthesize, the judgment of multiple experts to increase the calibration of the estimate used. Research has shown that a group of individuals assess an uncertain quantity better than the average expert, but the best individuals in the group are often better calibrated than the group as a whole [20]. The combination scheme used in this research is the classical model of Cooke [21]. Experience shows that Cooke's classical method outperforms both the best expert and the "equal weight" combination of estimates. In an evaluation involving 45 studies it performs significantly better than both in 27 studies and performs equally as well as the best expert in 15 of them [6].

In Cooke's classical method *calibration* and *information* scores are calculated for the experts based on their answers on a set of seed questions, i.e. questions for which the true answer is known at the time of analysis. The calibration score shows how well the respondent's answers represent the true value; the information score show how precise the respondent's answers are. These two scores are used to define a *decision maker* which assigns weights to the experts based on their performance. The weights defined by this decision maker are used to weight the respondents answer's to the questions of interest – in this case the effort estimates for vulnerability discovery projects. In sections 3.1, 3.2 and in 3.3 Cooke's classical method is explained. For a more detailed explanation the reader is referred to [21].

3.1 Calibration score

In the elicitation phase the experts provide individual answers to the seed questions. The seed questions request the respondents to specify a probability distribution for an uncertain continuous variable. This distribution is typically specified by stating its 5th, 50th, and 95th percentile values. These percentiles yield four intervals over the percentiles *[0-5, 5-50, 50-95, 95-100]* with probabilities of p= *[0.05, 0.45, 0.45, 0.05]*. As the seeds are realizations of these uncertain variables the well calibrated expert will have approximately 5% of the realizations in the first interval, 45 % of the realizations in the second interval, 45 % of the realizations in the third interval and 5% of the realizations in the fourth interval. If s is the distribution of the seeds over the intervals the relative information of s with respect to p is: $I(s,p) = \sum_{i=1}^{4} \ln(s_i / p_i)$.

This value indicates how surprised someone would be if one believed that the distribution was p and then learnt that it was s.

If N is the number of samples (seeds) the statistic of $2NI(s, p)$ is asymptotically Chi-square distribution with three degrees of freedom. This is asymptotic behavior is used to calculate the calibration *Cal* of expert *e* as: $Cal(e) = 1 - \chi_3^2(2N\,I(s,p))$. Calibration measures the statistical likelihood of a hypothesis. The hypothesis tested is that realizations of the seeds (*s*) are sampled independently from a distribution agreeing with the expert's assessments (*p*).

3.2 Information score

The second score used to weight experts is the information score, i.e. how informative the expert's distributions are. This score is calculated as the deviation of the expert's distribution to some meaningful background measure. In this study the background measure is a uniform distribution over [0,1].

If b_i is the background density for seed $i \in \{1,\dots,N\}$ and $d_{e,i}$ is the density of expert *e* on seed *i* the information score for expert *e* is calculated as: $inf(e) = \frac{1}{N}\sum_{i=1}^{N} I(d_{e,i}, b_i)$, i.e. as the relative information of the experts distribution with respect to the background measure. It should be noted that the information score does not reflect calibration and does not depend on the realization of the seed questions. So, regardless of what the correct answer is to a seed question a respondent will receive a low information score for an answer which is similar to the background measure, i.e. the answer is distributed evenly over the variable's range. Conversely, an answer which is more certain and has focused the probability density over few possible outcomes will yield high information scores.

3.3 Constructing a decision maker

The classical method rewards experts who produce answers with high calibration (high statistical likelihood) and high information value (low entropy). A strictly proper scoring rule is used to calculate the weights the decision maker should use. If the calibration score of the expert *e* is equal or greater than a threshold value the expert's weight is obtained as *w(e)=Cal(e)*Inf(e)*. If the expert's calibration is less than α the expert's weight is set to zero, a situation which is common to happen a substantial number of experts in practical applications.

The threshold value α corresponds to the significance level for rejection of the hypothesis that the expert is well calibrated. The value of α is identified by resolving the value that would optimize a virtual decision maker. This virtual decision maker combines the experts' answers (probability distributions) based on the weights they obtain at the chosen threshold value (α). The optimal level for α is where this virtual expert would receive the highest possible weight if it was added to the expert pool and had its calibration and information scored as the actual experts.

When α has been resolved the normalized value of the experts weights *w(e)* are used to combine their estimates of the uncertain quantities of interest.

4 Data collection method

This section presents how the data was collected in terms of: how seed questions for Cooke's classical method were constructed, the population and sample of experts that was chosen and how the elicitation instrument was developed and tested.

4.1 Seed questions

As the experts performance on answering the seed questions are used to weight them, it is critical that the seeds are well validated and also that they lie in the same domain as the studied variables. They need to be drawn from the respondents' domain of expertise, but need not necessarily be directly related to questions of the study [21].

Naturally, the robustness of the weights attributed to individual experts depends on the number of seeds used. This study used 11 seed questions. Experience shows this is more than enough to see substantial difference in calibration [21] between experts.

For this study two types of seed questions were used (cf. Table 2). All of these were constructed using information from the national vulnerability database and concerned characteristics of existing vulnerabilities in software products. Questions 1-5 concerned different types of vulnerabilities and under what conditions they could be exploited; questions 6-11 concerned how often publicly known vulnerabilities in different products was due to input validation or buffer errors and to authentication or authorization errors (cf. Table 3). Both these two types of questions are related to the topic as they gauge how well the expert can assess properties related to vulnerabilities that can be expected to be found.

Table 2. Seed questions used in abbreviated format and their realized value.

#	Question	Real
1	What portion of vulnerabilities published during 2010 of high severity has a complete impact on CIA	57 %
2	What portion of vulnerabilities published during 2010 of medium severity has a complete impact on CIA.	6 %
3	What portion of vulnerabilities published during 2010 that are remotely exploitable (does not require LAN access) will require that the attacker can authenticate itself before succeeding with an exploit?	9 %
4	What portion of vulnerabilities published in 2010 that are remotely exploitable (does not require LAN access) and requires that the attacker can authenticate itself before the exploit is of high severity?	15 %
5	What portion of vulnerabilities published in 2010 that are remotely exploitable (does not require LAN access) is of high severity?	52 %
6	What portion of vulnerabilities publicly announced in 2010 with high severity is due to input validation or buffer errors?	53 %
7	What portion of vulnerabilities publicly announced with high severity for Windows 7 is due to input validation or buffer errors?	36 %
8	What portion of vulnerabilities publicly announced with high severity for Apple's products is due to input validation or buffer errors?	31 %
9	What portion of vulnerabilities publicly announced with high severity for the .NET framework is due to authentication or authorization errors?	10 %
10	What portion of vulnerabilities publicly announced with high severity for the Microsoft's Internet Information Services is due to authentication or authorization errors?	13 %
11	What portion of vulnerabilities publicly announced with high severity for Cisco's products is due to authentication or authorization errors?	11 %

Table 3. Error types from NVD used.

Input validation/buffer errors	Authentication or authorization errors
CWE 20: Improper Input Validation	CWE 255: Credentials Management
CWE 89: SQL Injection	CWE 264: Permissions, Privileges, and Access Controls
CWE 119: Failure to Constrain Operations within the Bounds of a Memory Buffer	CWE 287: Improper Authentication
CWE 134: Uncontrolled Format String	CWE 310: Cryptographic Issues
CWE 189: Numeric Errors	

4.2 The domain experts

As this research aims to identify quantities related to discovery effort the respondents needed both the ability to evaluate aspects in the domain and the ability to reason in terms of probabilities. In terms of the expert categories described in [22] individuals that are expert judges are desirable.

Good candidates for this are researchers in the software security field. These can be expected to both understand how to reason with probabilities and to possess the required skills to evaluate the effectiveness difficulty of finding vulnerabilities in software. Software security researchers were therefore chosen as the population to survey. To identify suitable respondents, articles published in the SCOPUS database [23], INSPEC or Compendex [24] between January 2005 and September 2010 were reviewed. Authors was considered if they had written articles in the information technology field with any of the following phrases in the title, abstract or keywords: “software vulnerability”, “software vulnerabilities”, “software exploit”, ”software exploits”, “exploit development”, “develop exploits”„ “develop an exploit” ,”exploit writing”, “writing exploits”, "vulnerability research", or "exploit code". If their contact information could be found they were added to the sample of respondents. After reviewing and screening respondents and their contact information a sample of 384 individuals was assessed. The contact information for approximately 80 turned out to be incorrect or outdated.

As recommended by [25] , motivators were presented to the respondents invited to the survey: i) helping the research community as whole, ii) the possibility to win a gift certificate on literature, and iii) being able to compare their answers to other experts after the survey was completed. Out of approximately 300 researchers invited to the survey 92 opened the survey and 17 submitted answers to the survey's questions. A response rate of this magnitude is logically to be expected of a more advanced survey of this type.

4.3 Elicitation instrument

A web survey was used to collect the probability distributions from the invited respondents. The survey was structured into four parts, each beginning with a short introduction to the section. First, the respondents were given an introduction to the survey as such that explained the purpose of the survey and its outline. In this introduction they also confirmed that they were the person who had been invited and provided information about themselves, e.g. years of experience in the field of research. Second, the respondents received training regarding the answering format used in the survey. After confirming that this format was understood the respondents proceeded to its third part. In the third part both the seed questions and the questions of the study were presented to the respondents. Finally, the respondents were asked to provide qualitative feedback on the survey and the variables covered by it.

Questions in section 3 were each described through a scenario entailing a number of conditions. *Scenarios* and conditions for the seed questions can be found in Table 2; project types and conditions for the questions of interest in this study is described in section 2.1.

In the seed questions the respondent was asked to provide a probability distribution that expressed the respondent's belief. As is custom in applications of Cooke's classical method this probability distribution was specified by setting the 5th percentile, the 50th percentile (the median), and the 95th percentile for the probability distribution. In the survey the respondents specified their distribution by adjusting sliders or entering values to draw a dynamically updated graph over their probability distribution. The three points specified by the respondents defines four intervals over the range [0, 100]. The graphs displayed the probability density as a histogram, instantly updated upon change of the input values.

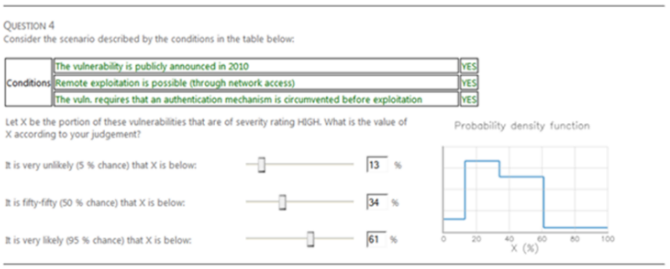
In the question of interest, the respondent specified probability distributions for work days required to find a zero-day vulnerability. The respondents were asked to specify the number of work days that would be needed to find a zero-day vulnerability with a probability of 5 percent, 50 percent and 95 percent. This is a common format to use for effort estimates [26] and in prediction in general [27]. As before the estimates dynamically updated a graph representing the answer. However, for these questions this graph showed the cumulative probability of finding a zero-day vulnerability as a function of work days spent. This graph was plotted using linear interpolation between the three values specified by the respondent.

Use of graphical formats is known to improve the accuracy of elicitation [28]. Figures and colors were also used to complement the textual formulations and make the content easier to understand. In Figure 7 the format presented to respondents is exemplified.

Elicitation of probability distributions is associated with a number of issues [28]. Effort was therefore spent on ensuring that the measurement instrument held sufficient quality. After careful construction the survey was qualitatively reviewed during personal sessions with an external respondent representative of the population. This session contained two parts. First the respondent was given the task to fill in the survey, given the same amount of information as someone doing it remotely.

After this discussions followed regarding the instrument quality. These sessions resulted in several improvements.

Before this qualitative review the question format as such had been tested in a pilot study on other security parameters. In that pilot study a randomized sample of 500 respondents was invited; 34 of these completed the pilot during the week it was open. The questions in this pilot survey were presented in the same way as in the present survey. A reliability test using Cronbach's alpha [29], [30] was carried out using four different ways to phrase questions for one variable. Results from this test showed α=0.817, which indicates good internal consistency of the instrument.



QUESTION 4

Consider the scenario described by the conditions in the table below:

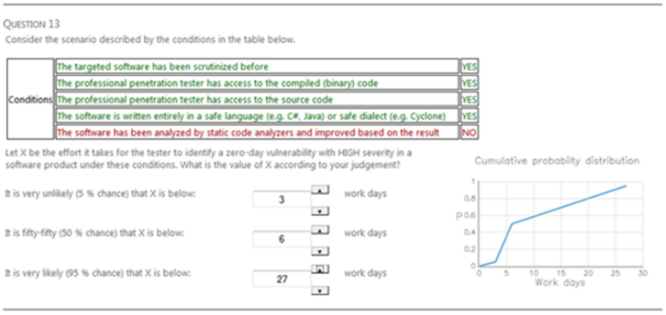
Conditions	The vulnerability is publicly announced in 2010	YES
	Remote exploitation is possible (through network access)	YES
	The vuln. requires that an authentication mechanism is circumvented before exploitation	YES

Let X be the portion of these vulnerabilities that are of severity rating HIGH. What is the value of X according to your judgement?

It is very unlikely (5 % chance) that X is below: 13 %

It is fifty-fifty (50 % chance) that X is below: 34 %

It is very likely (95 % chance) that X is below: 61 %



QUESTION 13

Consider the scenario described by the conditions in the table below.

Conditions	The targeted software has been scrutinized before	YES
	The professional penetration tester has access to the compiled (binary) code	YES
	The professional penetration tester has access to the source code	YES
	The software is written entirely in a safe language (e.g. C#, Java) or safe dialect (e.g. Cyclone)	YES
	The software has been analyzed by static code analyzers and improved based on the result	NO

Let X be the effort it takes for the tester to identify a zero-day vulnerability with HIGH severity in a software product under these conditions. What is the value of X according to your judgement?

It is very unlikely (5 % chance) that X is below: 3 work days

It is fifty-fifty (50 % chance) that X is below: 6 work days

It is very likely (95 % chance) that X is below: 27 work days

Figure 7. Examples of question and answering format in the survey (seed 4 and project type 2).

5 Results

This section presents the result of the analysis performed on the judgment of the 17 researchers. In section 5.1 the overall performance of the respondents on the seed questions is presented. In section 5.2 the synthesized estimates of those respondents who were assigned weight are presented. In section 5.3 the influence that each of the four individual variable have on the effectiveness is described.

5.1 Respondents' performance

As in many other studies involving expert judgment some of the respondents were poorly calibrated. Their calibration score varied between 0.540×10^{-3} and 0.615 with a mean of 0.305. The respondents' information score varied between 0.0770 and 1.009 with a mean of 0.324. Figure 8 shows the information score and calibration score of the 17 respondents.

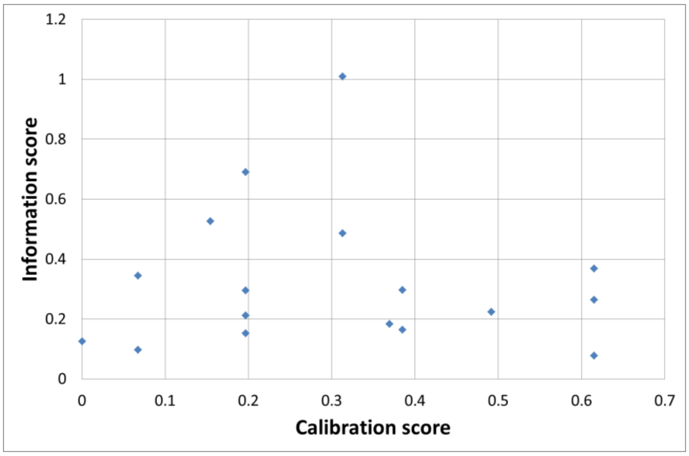


Figure 8. Information and calibration scores of the respondents.

Cooke's classical method aims is to identify those respondents whose judgment is well calibrated and informative. The virtual decision maker was optimized at a significance level (α) of 0.615. Consequently, the three rightmost respondents in Figure 8 received a weight higher than zero and the other 14 respondents received a weight of zero. As noted in section 3.3 above it is not uncommon that a substantial number of respondents receive the weight zero with this method.

The twelve respondents who received a positive weight all had the same calibration score (0.615). Their weights are therefore directly proportional with their information score (cf. section

3.2). They received weights 0.1086, 0.3711 and 0.5203 after normalization.

5.2 Work effort in the project types

To identify the probability distribution which the virtual decision maker assigns to the 16 types of vulnerability discovery projects examined the respondents' individual estimates were combined based on the respondent's weights. The estimated distributions were assumed to be distributed in the same way as they were presented to the respondents (c.f. section 4.3), i.e. as depicted in the linearly interpolated cumulative probability distributions for the finding of a zero-day vulnerability when work effort is increased.

The respondents specified the cumulative probability distribution through its 5th, 50th and 95th percentile. As depicted in Table 4 and the synthesized estimates show clear differences among the project types. The median for the projects varies between 1 and 14 work days; the value at the 5th percentile varies between 0 and 3 work days; the value at the 95th percentile varies between 7 and 855 work days. As could be expected is project type 5 the one with highest expected effort, closely followed by project type 7. For these two project types a time budget of more than 2 years and 4 months is needed to find a vulnerability with 95 percent certainty. In other project types this certainty can be obtained with a time-budget of just a week or a month. Project type 4, 10, 12 and 16 are associated with lowest work effort.

Table 1. Different types of vulnerability discovery projects and the estimated effort to find a vulnerability with a certain degree of certainty. Values have been rounded off to closest number of full days.

Project	Scrutinized	SourceCode	SafeLanguage	CodeAnalyzers	Low (5%)	Median(50%)	High95%)
1	Yes	Yes	Yes	Yes	3	13	74
2	Yes	Yes	Yes	No	1	3	26
3	Yes	Yes	No	Yes	0	13	26
4	Yes	Yes	No	No	0	1	7
5	Yes	No	Yes	Yes	1	12	855
6	Yes	No	Yes	No	0	10	27
7	Yes	No	No	Yes	2	9	855
8	Yes	No	No	No	1	4	257
9	No	Yes	Yes	Yes	1	6	27
10	No	Yes	Yes	No	0	4	9
11	No	Yes	No	Yes	0	3	17
12	No	Yes	No	No	1	3	8
13	No	No	Yes	Yes	1	14	344
14	No	No	Yes	No	1	7	27
15	No	No	No	Yes	1	6	18
16	No	No	No	No	0	3	9

5.3 Variables influence on the effectiveness

Four variables are varied to specify the 16 project types. The variation over scenarios supports this hypothesis that they influence effort. A relevant question is then how important these variables are for the effort required by the attacker. Table 5 shows the mean influence that the four variables have on the probability distribution. These values are the mean difference obtained when comparing scenarios where the variable is in the state true with those scenarios where the variable is in the state false, and all other variables remain in the same state. For instance, the values for *Scrutinized* are obtained as the mean value of the difference between scenarios 1 and 9, 2 and 10, 3 and 11 and so on.

All variables have a positive impact on the effort required to find a zero day vulnerability given a number of work days. As can be seen from Table 5 the most influential variables on the 95th percentile are *Scrutinized* (if the software has been searched for vulnerabilities before), *SourceCode* (if the attacker can get access to the source code) and *CodeAnalyzers* (if the software product has been improved with static code analyzers). The impact of these variables on the high extreme value, where a zero-day vulnerability is found with 95 percent probability, is substantial. Such sizeable difference cannot be found for the variable *SafeLanguage*. As a consequence this variable has a meager influence on the expected work effort in comparison to the other variables.

Table 5. Mean influence in work days of the variables under the assumptions used in the study.

Variable	Low (5%)	Median (50%)	High (95%)
Scrutinized	+0.4	+1.1	+208.5
SourceCode	+0.1	+3.6	+274.8
SafeLanguage	+0.4	+4.6	+24.0
CodeAnalyzers	+0.6	+3.9	+230.8

6 Discussion

Software insecurity is a serious problem in today's society. Decision makers can certainly make use of data on the effectiveness of measures that make vulnerability discovery projects more cumbersome. Most decision makers probably would prefer reliable empirical data to base their decisions on. However, such data is not available today. It is difficult to obtain such data from archival studies as no such archives are available and as indicated from the result of this study it would also be costly to collect this data from repeated experiments.

The use of expert judgment can be motivated in absence of reliable data. This study extracts and synthesizes data from domain experts. The method used to analyze the experts' judgments and combine these is described in section 6.1 below.

The elicitation instrument used is discussed in section 6.2. The result as such and the importance variables included in the study are discussed in section 6.3.

6.1 Expert judgment analysis

In this study Cooke's classical method [21] was used to synthesize expert judgments. This performance based method aims to select the experts that are well calibrated and combine their judgments in an optimal way. The track record of this method [6] positions it as the best-practice when it comes to combining experts' judgment of uncertain quantities.

Eleven seed questions were used to evaluate calibration and information scores. These seed questions are drawn from a vulnerability database. A concern to the validity is that this source also is available to the respondents who could have used them to identify the answers to the seed questions. If they would do so these seeds would not work well as a gauge for how well calibrated and informative the expert's own judgment is. However, it appears unlikely that anyone did so. None of the respondents answering the survey has given comments that indicate that they have realized that the correct answer can be found in online databases. Neither did the qualitative reviewer realize this during the qualitative reviews. Furthermore, inspections of the answers received do not indicate any answers were based on these sources.

The use of these seed questions shows that calibration varies among experts. This can be seen through the calibration scores to the seed questions used in this study (c.f. Figure 8). The three best calibrated experts were assigned weight when the virtual decision maker was optimized. The synthesized probability distributions created based on their judgment involve a great deal of uncertainty. In some cases the 95 percent confidence interval spans over 886 work days. As can be seen from Figure 8 , the estimates provided by the three respondents who obtained weight are not the most informative ones. This should not be seen as surprising. Overconfidence is a well-known cause for poor calibration in expert judgments [31]. Cooke's methods only assign weights to experts with a calibration score that exceeds a threshold value. However, these experts' weight is calculated with the information score as one of two factors. This avoids domination of uninformative experts in the synthesis of judgments.

When using this method it is appropriate to perform robustness test with respect to the seed variables and the experts by removing one expert and investigating the impact of this removal [21]. Such tests were performed and indicate that the solution is robust to changes in both seed questions and experts.

6.2 Validity and reliability of the elicitation instrument

Cooke [21] suggests that seven guidelines should be followed when data is elicited from experts. How these have been addressed in the present study is described below.

Cooke states that questions must be clear and unambiguous and that a dry run should be carried out before the actual study. In this study the clarity of questions were tested in qualitative reviews with a strategically selected respondent representative of the population. The comments received from this person helped improve the understandability of the instrument and remove ambiguity. Also, a quantitative test was performed on a survey with a similar structure and a similar way of phrasing questions. This quantitative test was made through a pilot survey answered by 34 respondents. It indicated good reliability of the survey instrument.

It is also suggested that an attractive graphical format and a brief explanation of the elicitation format should be prepared [21]. The answering format used in this study was supported by graphical illustrations – the answers were entered by entering a probability function on the screen. This format was also carefully explained in an introductory training section in the survey. Also, background information introduced each new section.

Cooke further recommends that the elicitation should not exceed one hour and that coaching should be avoided. None of the respondents who completed the survey spent more than one hour to do so and efforts were made to ensure that the questions were formulated in a neutral way.

The last recommendation given in [21] is that an analyst should be present when respondents answer the questions. The respondents were given contact information to the research group when invited to the survey and they were encouraged to use these any if questions arose. It is possible that analysts' physical absence from the elicitation suppressed some potential questions from being asked. In the survey the respondents were asked to comment the clarity of the questions and the question format used. Based on the comment received it appears as if the questions and the assumptions were understandable. Two respondents did however comment that the questions perhaps should be directed towards practitioners ("hackers") rather than researchers. While practitioners probably need more guidance in specifying answers through probability distributions this recommendation gives input to future research efforts in this track

6.3 Variables importance to zero-day discovery projects

Two weeks of work is enough to have a fifty-fifty chance of finding a zero-day vulnerability in all projects types assessed. In some cases two weeks of work is enough to give more than 95 percent chance of discovering a zero-day vulnerability and the fifty percent chance is reached after just a couple of days. While these estimates give dismaying results they are not in conflict with already known data. The rate with which vulnerabilities are publically announced hints that effort required to find them is modest. We also tested this prediction model using the PERT formula [32] on a number of software products which have been scrutinized. The estimates appear reasonable when compared to the publicly disclosed vulnerabilities in SecurityFocus [33] during 2010. For example, the estimates says that during all days of 2010 there would be the equivalent of approximately 7 professional 8 hour work-day on finding and disclosing vulnerabilities in Firefox, and that 17 professional penetration testers working each work-day on Internet Explorer 8.

No radical impact can be made using the measures included in this study, but they all help to increase the security of software products. Their impact on the median value is similar for all measures except making sure that products have been scrutinized (this has less impact on the median). The use of safe languages does not impact the extreme value (95th percentile) as much as the other ones. As a consequence, it does not influence the expected effort as much as the other three countermeasures do, and could be seen as less effective.

In the survey the respondents were asked to indicate if there were important variables missing. Only three out of 17 respondents suggested other priorities than used in the survey. All three suggested different things: fuzzers combined with static code analysis as one variable, if static code analysis was performed on a regular basis (not just performed), and variables indicating the security expertise of the developer and or development process (not specified which). All these suggestion were considered in the discussion with the panel of experts who prioritized variables to include in the survey, but were intentionally excluded from the survey. This, together with the survey-respondents' opinions indicates that the most important variables for estimating vulnerability discovery are included in this study.

While the most important variables seems to be included in our model the estimates indicate that the effort required to discover a new vulnerability can be as high as man-years even if the compiled code is available to the attacker. This study does not reveal which these conditions are, i.e. when the penetration tester will have to spend years searching for a vulnerability. The expert panel and the respondents of the survey indicated that the most important variables are included in the model used here. It is therefore likely that a number of favorable conditions must apply in these cases. In order to obtain better and more detailed knowledge in this area further work could explore what set of measures that causes this effect and how to achieve such secure software products.

7 Conclusion

It appears difficult to achieve a high level of security assurance in today's software intensive environment. The probability that a professional penetration tester will find a previously unknown vulnerability in software product used today is disturbingly high. Under most conditions a few days appears enough to find a zero-day vulnerability with a fifty percent chance. Countermeasures do increase work effort required, but none of them seem to have a striking impact on the effort required to find a vulnerability, at least not in the general case. The estimates made by experts included in this study are associated with a great deal of uncertainty. Under some conditions the professional penetration tester will need man years of effort required to find a zero-day vulnerability, i.e. the 95th percentile spans man-years. This study does not reveal which these conditions are, but since no crucial variables seem to be omitted from this study it is likely that a number of favorable conditions must apply in these cases.

8 References

[1] NIST Computer Security Resource Center (CSRC), "National Vulnerability Database," 2011. [Online]. Available: http://nvd.nist.gov/.

[2] O. H. Alhazmi and Y. K. Malaiya, "Quantitative vulnerability assessment of systems software," in *Proceedings of Annual Reliability and Maintainability Symposium*, 2005, pp. 615-620.

[3] S.-W. Woo, H. Joh, O. H. Alhazmi, and Y. K. Malaiya, "Modeling vulnerability discovery process in Apache and IIS HTTP servers," *Computers & Security*, vol. 30, no. 1, pp. 50-62, Jan. 2011.

[4] B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen, "On the secure software development process: CLASP, SDL and Touchpoints compared," *Information and Software Technology*, vol. 51, no. 7, pp. 1152-1171, Jul. 2009.

[5] A. Ozment, "Improving vulnerability discovery models," in *Proceedings of the 2007 ACM workshop on Quality of protection*, 2007, pp. 6–11.

[6] R. Cooke, "TU Delft expert judgment data base," *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May 2008.

[7] M. A. McQueen, T. A. McQueen, W. F. Boyer, and M. R. Chaffin, "Empirical estimates and observations of 0day

vulnerabilities," in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, 2009, pp. 1–12.

[8] C. Cowan, "Software security for open-source systems," *Security & Privacy, IEEE*, vol. 1, no. 1, pp. 38–45, 2003.

[9] M. Howard and D. C. LeBlanc, *Writing Secure Code*. Redmond, WA, USA: Microsoft Press, 2002.

[10] Y. Younan, "Efficient countermeasures for software vulnerabilities due to memory management errors," Katholieke Universiteit Leuven, 2008.

[11] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting vulnerable software components," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 529–540.

[12] S. Sridhar, K. Altinkemer, and J. Rees, "Software Vulnerabilities: Open Source versus Proprietary Software Security," *AMCIS 2005 Proceedings*, 2005.

[13] C. Payne, "On the security of open source software," *Information Systems Journal*, vol. 12, no. 1, pp. 61-78, Jan. 2002.

[14] T. Jim, G. Morrisett, D. Grossman, M. Hicks, J. Cheney, and Y. Wang, "Cyclone: A safe dialect of C," in *USENIX*, 2002, pp. 275-288.

[15] M. D. Penta, L. Cerulo, and L. Aversano, "The life and death of statically detected vulnerabilities: An empirical study," *Information and Software Technology*, vol. 51, no. 10, pp. 1469-1484, Oct. 2009.

[16] S. Heckman and L. Williams, "A systematic literature review of actionable alert identification techniques for automated static code analysis," *Information and Software Technology*, 2010.

[17] Y. Kim, J. Lee, H. Han, and K.-M. Choe, "Filtering false alarms of buffer overflow analysis using SMT solvers," *Information and Software Technology*, vol. 52, no. 2, pp. 210-219, Feb. 2010.

[18] S. Grimstad, M. Jorgensen, and K. Molokken-Ostvold, "Software effort estimation terminology: The tower of Babel," *Information and Software Technology*, vol. 48, no. 4, pp. 302-310, Apr. 2006.

[19] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Published by FIRST-Forum of Incident Response and Security Teams*, 2007, pp. 1-23.

[20] R. T. Clemen and R. L. Winkler, "Combining probability distributions from experts in risk analysis," *Risk Analysis*, vol. 19, no. 187, pp. 187-204, 1999.

[21] R. Cooke, *Experts in uncertainty: opinion and subjective probability in science*. 1991.

[22] D. J. Weiss and J. Shanteau, "Empirical Assessment of Expertise," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 1, pp. 104-116, 2003.

[23] Elsevier B.V., “Scopus,” 2011. [Online]. Available: http://www.scopus.com/.

[24] Elsevier Inc, “Engineering Village,” 2011. [Online]. Available: http://www.engineeringvillage.com. [Accessed: 24-Feb-2011].

[25] S. T. Cavusgil and L. A. Elvey-Kirk, “Mail survey response behavior: A conceptualization of motivating factors and an empirical study,” *European Journal of Marketing*, vol. 32, no. 11/12, pp. 1165–1192, 1998.

[26] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*, 7th ed. New York, NY, USA: John Wiley & Sons, 2001.

[27] J. Armstrong, *Principles of forecasting – A Handbook for Researchers and Practitioners*. Netherlands: Kluwer Academic Publishers Group, 2001.

[28] P. H. Garthwaite, J. B. Kadane, and A. O’Hagan, “Statistical methods for eliciting probability distributions,” *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 680-701, 2005.

[29] L. J. Cronbach and R. J. Shavelson, “My Current Thoughts on Coefficient Alpha and Successor Procedures,” *Educational and Psychological Measurement*, vol. 64, no. 3, pp. 391-418, Jun. 2004.

[30] L. J. Cronbach, “Coefficient alpha and the internal structure of tests,” *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[31] S. Lin, “A study of expert overconfidence,” *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 711-721, May 2008.

[32] H. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*, 7th ed. New York, NY, USA: John Wiley & Sons, 2001.

[33] SecurityFocus, “SecurityFocus,” 2011. [Online]. Available: http://www.securityfocus.com/. [Accessed: 25-Feb-2011].

Paper C: Estimates of success rates of remote arbitrary code execution attacks

Teodor Sommestad, Hannes Holm and Mathias Ekstedt

Abstract

Purpose: To identify the importance of the factors that influence the success rate of remote arbitrary code execution attacks. In other words, attacks which use software vulnerabilities to execute the attacker's own code on targeted machines. Both attacks against servers and attacks against clients are studied.

Design/methodology/approach: The success rates of attacks are assessed for 24 scenarios: 16 scenarios for server-side attacks and 8 for client-side attacks. The assessment is made through domain experts and is synthesized using Cooke's classical method, an established method for weighting experts' judgments. The variables included in the study were selected based on the literature, a pilot study, and interviews with domain experts.

Findings: Depending on the scenario in question, the expected success rate varies between 15 and 67 percent for server-side attacks and between 43 and 67 percent for client-side attacks. Based on these scenarios, the influence of different protective measures is identified.

Practical implications: The results of this study offer guidance to decision-makers on how to best secure their assets against remote code execution attacks. These results also indicate the overall risk posed by this type of attack.

Originality/value: Attacks that use software vulnerabilities to execute code on targeted machines are common and pose a serious risk to most enterprises. However, there are no quantitative data on how difficult such attacks are to execute or on how effective security measures are against them. This study provides such data using a structured technique to combine expert judgments.

1 Introduction

The presence of software vulnerabilities in information systems is an important source of risk. Software vulnerabilities can be exploited by adversaries to gain access to sensitive information, to abuse functionality or to consume other system resources. In some cases, it is possible to remove a vulnerability by applying a software patch. In other cases, this type of removal is not possible, either because the vendor has not issued such a patch or because the vendor and the public are unaware of the vulnerability's existence. Also, in many cases, the cost or risk associated with applying a patch (e.g., the service being unavailable during the patching process) hinders the management from applying the patch in a timely fashion.

Software vulnerabilities that can be used to obtain remote control over a machine belong to the most severe examples. Such vulnerabilities are typically exploited by injecting malicious instructions into the memory space of the software that is running on the targeted machine and passes control of the system to the attacker. They are collectively called "arbitrary code vulnerabilities" and include buffer overflow vulnerabilities, dangling pointer references, insecure use of format strings, and integer errors [1].

The risk that an organization faces when such vulnerabilities are present in one of their systems is contingent on the probability that the vulnerabilities can be successfully exploited in practice. Some vulnerabilities are by nature more difficult to exploit than others, and it is possible to apply a number of security measures that makes exploitation more difficult [1].

Because the risk that an organization faces is highly dependent on the likelihood of successful exploitation, data regarding this aspect are very valuable when performing risk analysis, e.g., of a specific vulnerability or when using attack graph approaches such as [2–5]. However, data on the likelihood of successful exploitation are difficult to obtain because there are many relevant factors for the success of the exploitation. To generalize from observations would require tests on representative samples of vulnerabilities in different environments, with various security measures in place, and involving attackers who are representative of some category of adversary. Thus, it is immensely expensive to gain sound results through experiments, and as a consequence, they are rarely performed. The few experiments that have been performed on the subject have successfully demonstrated technical limitations of measures used in isolation, but they have not reported the difficulty of exceeding these limitations in practice. For example, Shacham et al. [6] tested the effectiveness of address space layout randomization under certain conditions but do not show how often these conditions apply in practice. Wilander and Kamkar [7] performed tests of a few protective measures against buffer overflows of different forms. However, without data on the attack forms used in practice, it is difficult to derive useful success rates from this data. Many of the tests that have been performed are of low relevance to practitioners (e.g., network administrators) because they evaluate defense mechanisms that are very difficult to implement, for example, because they are not supported by common operating systems.

Expert judgment is often used when quantitative data are difficult to obtain from experimental studies or by other means. Expert judgment, for example, has been used to assess the importance of attributes that are related to critical infrastructure risks [8] and to quantify parameters in security risk models [9]. This paper describes a study in which expert judgment was used to quantify the success rate of remote arbitrary code execution attacks in 24 different attack scenarios.

An important issue when eliciting expert judgment is that of bias. In other words, experts are prone to various types of bias, e.g., relating to their background. This study synthesizes the judgment of 21 domain experts using an established performance-based method known as Cooke's classical method [10]. This method assigns weights to domain experts' judgments based on their ability to estimate the true value for a number of seed questions, that is, questions related to the subject matter and for which the true answer is known. These seed questions are used to identify experts who are suitable to answer the questions of interest, i.e., experts who have both the relevant background knowledge and the ability to express their knowledge quantitatively. Seed questions in this study were designed to find experts that are suitable for estimating the success rate of remote arbitrary code exploits. The experts' performance on these seed questions are used to weight their assessments of the 24 attack scenarios.

The 21 domain experts assessed 16 scenarios related to server-side attacks and 8 scenarios related to client-side attacks. These scenarios are used to analyze the effectiveness of the various defense mechanisms that have value for network administrators or decision-makers in security issues. The uncertainty of these estimates is also described. Both the research method used and the variables included in the scenarios have been previously tested in a pilot study [11].

2 Attack scenarios

This study quantifies the probability that remote arbitrary code execution attacks will succeed given that they are executed. Many variables influence whether such an attack succeeds or not. The presence of a software vulnerability that enables the execution of arbitrary code is a necessary condition (e.g., a buffer overflow vulnerability [12]). Some vulnerabilities are only exploitable under certain conditions. Two variables that are often used to describe when a software vulnerability can be exploited are (1) whether the vulnerability can be exploited remotely or locally and (2) whether the attacker would need to bypass some authentication mechanism before the vulnerability can be exploited [13].

Furthermore, countermeasures against code execution attacks can be deployed both on a network and a machine level. Deep-packet-inspection firewalls and filtering proxies are two network-based measures that can prevent the executable code from reaching its target [14]. Measures that are deployed on a machine level include [1] non-executable memory protection (NX), which makes certain parts of memory impossible to use in executing code, guard page-based countermeasures that terminate programs that access certain parts of memory, execution monitors that execute programs in a "sandbox" or that search for anomalies in execution, address space layout randomizations (ASLR), which obfuscate the memory for the attacker, and instruction set randomizations that encrypt the program instructions so that attackers cannot insert their own instructions without the decryption key.

All of these protective measures have multiple implementations and variants that are available, for example, for different operating system platforms. However, for a variety of reasons, they are not all used in practice. Because the aim of this research is to construct a model that is useful for enterprise decision makers, such as network administrators, the focus is placed on variables that are common in practice. The list of chosen variables was assessed by using the following: 1) literature studies, 2) a pilot study [11], and 3) three interviews with respondents who had significant practical experience from arbitrary code attacks.

The chosen variables include the protection mechanisms *NX* and *ASLR,* which are straightforward to turn on or off in commonly used operating systems. Deep-packet-inspection firewalls (*DPI*) and filtering proxies (*Proxy*) are also included as variables. The latter two are also common in today's enterprise environments. Additionally, in server-side attacks, it is significant if the attacker can authenticate himself as a legitimate user [14]. Because the existence of authentication mechanisms is something that can be influenced in practice, it is included as the variable *AccessControl* in the attack scenarios. *Connectivity* and *CraftResponse* can be seen as necessary (but not sufficient) conditions for a remote attack. If *Connectivity* is true, then the attacker can connect to the service that is to be attacked; if *CraftResponse* is true, then the attacker can create data that are fed to the client. In practice, the presence of a high-severity vulnerability (*Vulnerable*) is also a necessary condition for remote code execution.

Naturally, the attacker's competence and resources are also of importance to the probability of successful remote code execution. Such attacker properties are kept constant for all of the studied attack scenarios (cf. Table 1). In other words, the attacker is a professional penetration tester who has access to open and commercially available tools and has one week of time to prepare for the attack (e.g., to probe the host and tune the exploit).

Table 1 and Figure 1 summarize all of the variables as well as the states of these variables, which were used to create the scenarios. The overall hypotheses are that that those variables varied over the scenarios significantly influence the probability of success in remote arbitrary code execution attacks, and that this model is well-suited for predictions of success in remote arbitrary code execution attacks.

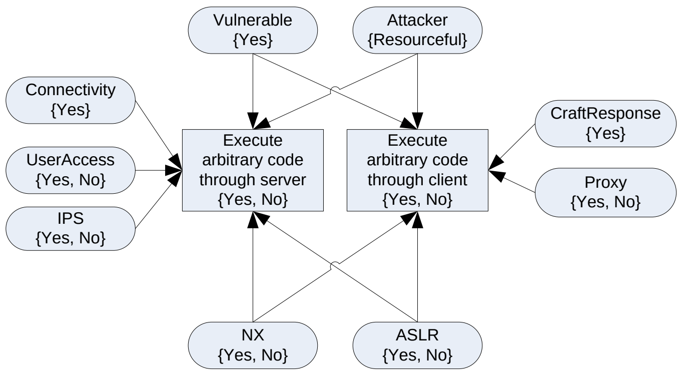


Figure 1. Variables that was included in the attack scenarios and dependencies that were investigated.

Table 2. Variables included in the attack scenarios.

Variable	States studied	Description
Proxy	Yes/No	If a filtering proxy, e.g. a filtering web proxy, is between the attacker server and the client.
DPI	Yes/No	If a deep-packet-inspection firewall is located between the attacker and the targeted server.
AccessControl	Yes/No	If the attacker can authenticate itself as a legitimate user of the service that is exploited in the attack. E.g., this variable is true if the attacked service is the SMB service (file and printer sharing) and the attacker is a part of the service's windows domain.
NX	Yes/No	If non-executable memory protection is activated on the targeted machine and used for the service attacked, e.g., DEP on a Windows machine or PaX on a Linux machine.
ASLR	Yes/No	If address space layout randomization is activated on the targeted machine.
Vulnerable	Yes	The targeted software has a high-severity vulnerability (as defined by CVSS [13]).
Connectivity	Yes	The attacker can send requests to the targeted service, e.g. because the firewall allows such connections.
CraftResponse	Yes	The attacker can craft (malicious) responses to the client, e.g., by luring the user of a web browser to a website controlled by the attacker.
Attacker	Resource ful	The attacker is a professional penetration tester with access to open and commercially available tools, and with one week to prepare the attack.

3 Synthesizing expert judgments

There is a substantial amount of research on how to combine, or synthesize, the judgment of multiple experts to increase the calibration of the estimates used. These techniques include the following: consensus methods [15], [16], the Cochran-Weiss-Shanteau index [17], self-proclaimed expertise [18], experience [19], certifications [19], peer-recommendations [19], and Cooke's classical method [10]. There is little research that compares the accuracy that these methods yield. However, research has shown that groups of individuals assess an uncertain quantity better than the average expert, while the best individuals in the group are often better calibrated than the group as a whole [20]. The scheme used to combine judgments in this research is the one used in the classical model of Cooke [21]. Cooke's model is a generic method for combining expert judgments that has been applied to a number of different domains. Experience from applications of Cooke's classical method has shown that it outperforms both the best expert and the "equal weight" combination of estimates. In an evaluation involving 45 studies, it performed significantly better than both alternatives in 27 studies and equally well as the best expert in 15 of the studies [22].

In Cooke's classical method, *calibration* and *information* scores are calculated for the experts based on their answers to a set of seed questions, i.e., questions for which the true answer is known at the time of analysis. These two scores are used to define a *decision maker* that assigns weights to the experts based on their performance. These weights are used to create a single estimate on the variables of interest – in this case, the 24 attack scenarios. Cooke's classical method is briefly explained in Sections 3.1, 3.2 and 3.3. The reader is referred to [21] for a detailed explanation of the method.

3.1 Calibration score

In the elicitation phase, the experts provide individual answers to the seed questions. The seed questions request that the respondents specify a probability distribution for a continuous variable for which the true value is uncertain to the respondent. This distribution is typically specified by stating its 5th, 50th, and 95th percentile values. This set of values yields four intervals over the percentiles *[0-5,5-50,50-95,95-100]* with probabilities of *p=[0.05,0.45,0.45,0.05]*. Because the seeds are realizations of these variables, a well-calibrated expert will have approximately 5% of the realizations in the first interval, 45% of the realizations in the second interval, 45% of the realizations in the third interval and 5% of the realizations in the fourth interval. If *s* is the distribution of the seed over the intervals, then the relative information of *s* with respect to *p* is the following: $I(s,p) = \sum_{i=1}^{4} \ln(s_i/p_i)$. This value indicates how surprised someone would be if one believed that the distribution was *p* and then learned that it was *s*.

If N is the number of samples/seeds, the statistic of 2*NI*(*s*, *p*) is asymptotically Chi-square distributed with three degrees of freedom. This asymptotic behavior is used to calculate the calibration (*Cal)* of expert *e* as the following: $Cal(e) = 1 - \chi_3^2(2N\, I(s,p))$. The calibration measures the statistical likelihood of a hypothesis. The hypothesis tested is that realizations of the seeds (*s*) are sampled independently from distributions that agree with the expert's assessments (*p*).

3.2 Information score

The second score used to weight experts is the information score, i.e., how precise and informative the expert's distributions are. This score is calculated as the deviation of the expert's distribution from some meaningful background measure. In this study, the background measure is a uniform distribution over [0,1].

If b_i is the background density for seed i∈{*1,…,N*} and $d_{e,i}$ is the density of expert *e* on seed *i*, the information score for expert *e* is calculated as the following: $\inf(e) = 1/N \sum_{i=1}^{N} I(d_{e,i}, b_i)$, which is the relative information of the experts' distribution with respect to the background measure.

3.3 Constructing a decision maker

The classical method rewards experts who produce answers that have a high calibration (high statistical likelihood) and a high information value (low entropy). A strictly proper scoring rule is used to calculate the weights of the decision maker. If the calibration score of the expert *e* is at least as high as a threshold value, then the expert's weight is obtained as the following: *w(e)=Cal(e)*Inf(e)*. If the expert's calibration is less than the threshold value, the expert's weight is set to zero, a situation that is common in practical applications.

The threshold value corresponds to the significance level for the rejection of the hypothesis that the expert is well-calibrated. This value is the value that would optimize a virtual decision maker if it were added to the expert pool and had its weight calculated as one of the actual experts. When the threshold value is resolved, the normalized value of the expert weights *w(e)* is used to combine their estimates of the uncertain quantities of interest.

4 Method

4.1 Seed questions

Since the experts' performance in answering the seed questions is used to weight the experts, it is critical that the seeds are correct and are in the same domain as the variables that are studied. They need to be drawn from the relevant domain of expertise but do not need to be directly related to questions of the study [21].

Naturally, the robustness of the weights that are given to individual experts depends on the number of seeds used. Experience shows that eleven seed questions are more than sufficient to see substantial differences in calibration [21].

Two types of seed questions were used in this study. For the first type, questions (cf. #1-5 in Table 3) were drawn from the National Vulnerability Database (NVD) [23] and concern statistics on known vulnerabilities in software products. The second type of question concerns the effectiveness of protective measures for buffer overflow vulnerabilities and was taken from [7]. Questions of the second type (cf. #5-11 in Table 3) asked the respondents to estimate how efficient protective measures were against 20 forms of attack that were described together with the questions.

Table 3. Seed questions and their realization values.

#	Question summary	Realization (%)
1	How many of the high-severity vulnerabilities published in 2010 have a full impact on Confidentiality, Integrity and Availability?	57
2	How many of the medium-severity vulnerabilities published in 2010 have a full impact on Confidentiality, Integrity and Availability?	6
3	How many of the vulnerabilities published in 2010 that can be exploited remotely require that the attacker bypass some authentication mechanism first?	9
4	How many of the vulnerabilities published in 2010 that can be exploited remotely and require that the attacker bypass some authentication mechanism first is of severity-rating high?	15
5	How many of the vulnerabilities published in 2010 that can be exploited remotely are of severity-rating high?	52
6	What is the probability that an attack (selected randomly from the 20 listed) will be prevented if Libverify and Libsafe are used?	0
7	What is the probability that an attack (selected randomly from the 20 listed) will be halted if Libverify and Libsafe are used?	20
8	What is the probability that an attack (selected randomly from the 20 listed) will be prevented if ProPolice is used?	40
9	What is the probability that an attack (selected randomly from the 20 listed) will be halted if ProPolice is used?	10
10	What is the probability that an attack (selected randomly from the 20 listed) will be prevented if Stackguard's terminator canary is used?	0
11	What is the probability that an attack (selected randomly from the 20 listed) will be halted if Stackguard's terminator canary is used?	15

4.2 The domain experts

Studies of expert calibrations have concluded that experts are well-calibrated in situations with learnability and with ecological validity [24]. Learnability is facilitated by the existence of models of the domain of interest; the possibility of expressing judgments in a coherent and quantifiable manner that can be verified; and the opportunity to learn from historic predictions and outcomes. Ecological validity is present if the expert is used to make judgments of the type that are requested.

In the context of this study, the above reasoning implies that good candidates are researchers and penetration testers in the security field. These individuals can be expected to be experienced in reasoning about the success or failure of attacks under different conditions and are expected to observe the outcomes of attempts. They also make judgments in their line of work (i.e., provide ecological validity).

To identify suitable respondents, articles published in the SCOPUS database [25], INSPEC or Compendex [26] between January 2005 and September 2010 were reviewed. Authors who had written articles in the information technology field with any of the words: "remote code execution", "run arbitrary code", "execute arbitrary code", "arbitrary code execution", "buffer overflow", "buffer overrun" or "exploit code" in the title, abstract or keywords were identified. If their contact information could be found, they were added to the list of potential respondents, resulting in a sample of 964 individuals.

After the exclusion of individuals for which no contact information could be found and a manual review of their publications' topicality, a sample of 545 individuals was assessed. Contact information for approximately 110 of these individuals turned out to be incorrect or outdated, resulting in approximately 445 invitations reaching their destination.

A web survey was conducted during five weeks in December 2010 to January 2011. Out of approximately 445 researchers who were invited to take the survey, 119 opened it and 19 submitted answers to the survey's questions. A response rate of this magnitude is to be expected of an advanced survey such as this one. As recommended by [27], motivators were presented to the respondents invited to the survey: i) helping the research community as whole, ii) the possibility to win a gift certificate for academic literature, and iii) being able to compare their answers to other experts after the survey was completed. One respondent provided contradictory and incomplete answers to the questions. After being unsuccessful in confirming these answers with this respondent, the respondent was excluded from further analysis, resulting in 18 usable surveys from researchers.

Additionally, practitioners were identified based on peer recommendations from notable practitioners in Sweden. Three practitioners, all with substantial experience in security exploits, participated in the study. Because practitioners are less likely to be as familiar with questionnaires in general and probability density functions in particular, these three respondents were given instructions on how to answer the survey during personal meetings in February and March 2011. Apart from the personal meetings, the participating practitioners answered the questionnaire in the same manner as the invited sample of researchers.

Thus, together with the three practitioners' surveys, the total number of respondents was 21.

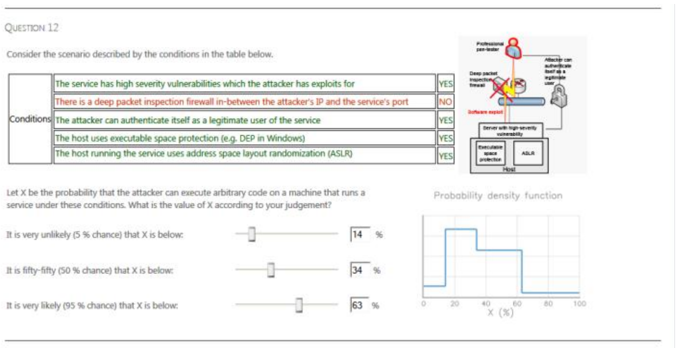
4.3 Elicitation instrument

The web survey comprised four parts, each beginning with a short introduction to the section. First, the respondents were given an introduction to the survey that explained the purpose of the survey and its outline. In this introduction, they confirmed that they were the person who had been invited and provided information about themselves, e.g., their number of years of experience in the field of research. Second, the respondents received training regarding the answering format used in the survey. After confirming that this format was understood, the respondents proceeded to its third part. Third, both the seed questions and the questions of the study were presented to the respondents. Finally, the respondents were asked to provide qualitative feedback on the survey and the variables that it covered.

Questions in section 3 of the survey were described through scenarios that detailed conditions for an attack. Summaries of the scenarios in the seed questions can be found in Table 1; conditions for the scenarios of interest in this study are described in section 2 of the paper.

For each scenario, the respondent was asked to provide a probability distribution that expressed the respondent's belief. As is customary in applications of Cooke's classical method (cf. Section 3), this probability distribution was specified by setting the 5th percentile, the 50th percentile (the median), and the 95th percentile for the probability distribution. In the survey, the respondents specified their distribution by adjusting sliders or entering values to draw a dynamically updated graph over their probability distributions. The three points specified by the respondents defined four intervals over the range [0, 100]. The use of graphical formats is known to improve the accuracy of elicitation [28]. Figures and colors were also used to complement the textual questions and to make the questions easier to understand. In Figure 2, the format presented to respondents is exemplified. A larger, generic figure that described the survey's variables could also be found at the top of each section, along with introductory text.

Elicitation of probability distributions is associated with a number of issues [28]. Efforts were therefore made to ensure that the measurement instrument was of sufficient quality. After careful construction, the survey was qualitatively reviewed during a personal session with an external respondent representative of the population. This session was divided into two parts. First, the respondent was given the task of filling in the survey, given the same amount of information as someone doing it remotely. After this task, discussions followed regarding the instrument quality. The qualitative review resulted in some minor improvements with respect to the phrasing of questions.


QUESTION 12

Consider the scenario described by the conditions in the table below.

Conditions	The service has high severity vulnerabilities which the attacker has exploits for	YES
	There is a deep packet inspection firewall in-between the attacker's IP and the service's port	NO
	The attacker can authenticate itself as a legitimate user of the service	YES
	The host uses executable space protection (e.g. DEP in Windows)	YES
	The host running the service uses address space layout randomization (ASLR)	YES

Let X be the probability that the attacker can execute arbitrary code on a machine that runs a service under these conditions. What is the value of X according to your judgement?

It is very unlikely (5 % chance) that X is below: 14 %

It is fifty-fifty (50 % chance) that X is below: 34 %

It is very likely (95 % chance) that X is below: 63 %

Professional pen-tester
Attacker can authenticate itself as a legitimate user
Deep packet inspection firewall
Software exploit
Server with high-severity vulnerability
Executable space protection
ASLR
Host
Probability density function
0
20
40
60
80
100
X (%)

Figure 2. Example of the question-and-answer format of the survey.

Before this qualitative review, the question format had been tested in a pilot study on other security parameters. In that pilot study, a randomized sample of 500 respondents was invited; 34 of these respondents completed the pilot during the week it was open. The questions in this pilot survey were presented in the same way as in the present survey. A reliability test using Cronbach's alpha (Cronbach and Shavelson, 2004; Cronbach, 1951) was performed using four different ways to phrase the questions for one variable. Results from this test showed a reliability value (alpha) of 0.817, which indicated good internal consistency of the instrument.

5 Results

5.1 Respondents' performance

As in many other studies that involve expert judgment, many of the experts were poorly calibrated on the seed questions. Their calibration scores varied between $3.211*10^{-14}$ and 0.6362, with a mean of 0.004255, and their information scores varied between 0.0658 and 1.847, with a mean of 0.7879.

Cooke's classical method aims to identify those respondents whose judgment is well calibrated and informative. The virtual decision maker was optimized at a threshold level (significance level) of 0.0007985. Four experts passed this threshold level and were assigned weights. They received the weights 0.8459, 0.1279, 0.02483, and 0.001361 after normalization. All four were researchers; their average experience from research on arbitrary code attacks was 12 years. As noted in Section 3.3, it is not uncommon that a substantial number of respondents receive a weight of zero with this method.

5.2 Success rates of arbitrary code execution attacks

The respondents' weights were used to construct the estimates of the virtual decision maker's estimates of success rates. In other words, the estimates described in this section represent the estimate of a virtual expert that is obtained by weighting the individual estimates of the respondents according to Cooke's method. The estimated distributions were assumed to be distributed in the same way that they were presented to the respondents, i.e., as depicted in the histograms over the four ranges that they constructed with their answers (c.f. Section 4.3). Note that certain variables are kept constant over the scenarios (c.f. Section 2).

5.2.1 Server-side attacks

As depicted in Table 3, the synthesized estimates show clear differences among the scenarios. The median for the scenarios varies between 10 and 75 percent; the value at the 5th percentile varies between 1 and 17 percent, and the value at the 95th percentile varies between 48 and 94 percent. As one might expect, scenario 1 has the lowest median (10%) and expected (15%) success rate. Scenario 16 has, as one might expect, the highest success rate.

Table 4. Attack scenarios for server-side attacks.

Scenario	Access Control	DPI	NX	ASLR	Low (5%)	Median (50%)	High (95%)	Expected (Mean)
1	Yes	Yes	Yes	Yes	1	10	51	15
2	Yes	Yes	Yes	No	4	15	60	20
3	Yes	Yes	No	Yes	6	20	62	24
4	Yes	Yes	No	No	6	26	69	32
5	Yes	No	Yes	Yes	4	21	48	24
6	Yes	No	Yes	No	4	25	56	27
7	Yes	No	No	Yes	4	30	63	33
8	Yes	No	No	No	5	41	86	43
9	No	Yes	Yes	Yes	7	36	79	41
10	No	Yes	Yes	No	7	38	79	41
11	No	Yes	No	Yes	5	27	68	31
12	No	Yes	No	No	14	69	94	65
13	No	No	Yes	Yes	11	45	88	48
14	No	No	Yes	No	14	66	89	59
15	No	No	No	Yes	15	50	89	52
16	No	No	No	No	17	75	94	67

5.2.2 Client-side attacks

Table 5 lists the virtual decision maker's estimates for the eight attack scenarios considered for client-side attacks. In terms of the expected success rate, the difference between the most secure scenario (#17) and the least secure scenario (#24) is 24 percentiles. The low success rates associated with the server-side attacks where the attacker cannot gain user access is not present in these scenarios – the data received by the client are implicitly trusted by it.

Table 5. Attack scenarios for client-side attacks.

Scenario	Proxy	NX	ASLR	Low (5%)	Median (50%)	High (95%)	Expected (Mean)
17	Yes	Yes	Yes	7	38	84	43
18	Yes	Yes	No	10	43	89	47
19	Yes	No	Yes	12	48	94	52
20	Yes	No	No	15	53	94	55
21	No	Yes	Yes	4	54	95	56
22	No	Yes	No	15	58	94	59
23	No	No	Yes	18	63	95	62
24	No	No	No	20	72	95	67

5.3 Variables' influence on the success rate of exploits

This study varies four variables in each set of scenarios. The variation over the scenarios supports the hypothesis that these variables are relevant for the success rate. Table 5 shows their mean influence on the estimates. These values are the mean difference obtained when comparing scenarios in which the variable is in the state of "true" with those scenarios in which the variable is in the state "false" and all other variables remain in the same state. For example, the values for *AccessControl* in the server-side scenarios are obtained as the mean value of the difference between the following scenarios: 1 and 9; 2 and 10; 3 and 11; and so on. A combination of variables (e.g., "DPI & NX") shows the mean influence that the combination has when compared to the individual influences that they have alone. A positive value for a combination indicates that the measures cancel each other out to an extent; a negative value indicates that the combined measures complement each other and that the joint effect is greater than the sum of the individual measures.

As can be seen from Table 5, restriction of access influences server-side attacks the most wheras the presence of a filtering proxy shows the most influence on client-side attacks. The respondents seem to perceive the studied variables as fairly independent, i.e., the effects from combinations of them are small.

Table 6. Mean influence of the variables on the success rate (in percent).

Scenario	Variable	Low (5%)	Median (50%)	High (95%)	Expected (Mean)
Server	AccessControl	-7.00	-27.25	-23.13	-23.25
	DPI	-3.00	-14.00	-6.38	-10.50
	NX	-2.50	-10.25	-9.38	-9.00
	ASLR	-2.25	-14.50	-9.88	-10.75
	AccessControl & DPI	+3.00	+2.50	+3.63	+1.50
	AccessControl & NX	+0.50	-1.25	-6.88	-2.50
	AccessControl & ASLR	+1.25	+8.00	-1.88	+4.25
	DPI & NX	-0.50	-0.50	+3.38	+0.25
	DPI & ASLR	-0.75	+0.75	-0.63	-1.00
	AccessControl & DPI & NX	-1.00	+1.50	+2.88	+0.75
	AccessControl & DPI & ASLR	+0.25	+0.25	+4.38	+1.00
	DPI & NX & ASLR	+0.75	+3.75	+0.63	+3.25
	AccessControl & DPI & NX & ASLR	-1.75	-5.25	-4.88	-4.25
Client	Proxy	-3.25	-16.25	-4.50	-11.75
	NX	-7.25	-10.75	-4.00	-7.75
	ASLR	-4.75	-5.75	-1.00	-3.75
	Proxy & NX	+2.25	+0.75	-3.50	-0.75
	Proxy & ASLR	+1.75	+0.75	-1.50	+0.25
	NX & ASLR	-2.25	+1.25	-1.0	+0.25
	Proxy & NX & ASLR	+2.25	-1.25	-1.5	-0.75

6 Discussion

6.1 The expert judgment analysis

Eleven seed questions were used to evaluate the calibration and information scores. These seed questions are of two types. The first type of seed question is drawn from a vulnerability database and concerns the characteristics of known vulnerabilities. The second type is drawn from an empirical peer-reviewed study [7] on the types of exploits that different countermeasures protect against. Both of these types of questions are strongly related to the expertise that is required to answer the question of interest. A concern about the survey's validity could be that these sources are available to the respondents, who could have used them to identify the answers to the seed questions. However, no indications of this concern were seen in the answers received or in the feedback from the respondents.

The calibration scores show that many experts in the field are poorly calibrated, i.e., their estimates do not match empirical observations well. This observation suggests that sorting out well-calibrated experts is worthwhile. Four respondents were assigned weights when the virtual decision-maker was optimized. When using this method to assign weights, it is appropriate to perform a robustness test on the solution [21]. These tests are performed with respect to both seed variables and experts by removing one at a time and by investigating the impact of the omission [21]. Such tests were performed and no undue influence was identified.

6.2 Validity and reliability of the elicitation instrument

Cooke [21] suggests that seven guidelines should be used when data are elicited from experts: i) formulate clear questions, ii) use an attractive format for the questions and a graphical format for the answers, iii) perform a dry run, iv) have an analyst present during the elicitation, v) prepare an explanation of the elicitation format and how answers will be processed, vi) avoid coaching and vii) keep elicitation sessions to less than one hour long.

This study follows all of these guidelines except for iv), which is to have an analyst present during the elicitation. The invited researchers were given contact information to the research group when invited to the survey, which they were encouraged to use if any questions arose. Practitioners were also introduced to the survey format personally. However, it is possible that the physical absence of the analysts suppressed some potential issues from being brought up during the elicitation. In the survey, the respondents were asked to comment on the clarity of the questions and the question format used. Based on the comments received, it appears as though the questions and the assumptions were fully understood.

6.3 Variables of importance to the success rate

The models used to describe attack scenarios in this study contained four variables for server-side attacks and three variables for client-side attacks. All these variables have an influence on the success rate. The result shows that the most influential countermeasures against server-side attacks are to make certain that attackers do not obtain access credentials to the service. If the attacker does not have access rights for the service, the expected success rate is decreased by 23 percentiles on average. However, restricting access can be difficult, for example, in the case of public services. Address space layout randomization, non-executable memory, and deep-packet inspection also lower the attack success rate significantly. Taken together, these three countermeasures lower the expected success rate by 26-28 percentiles. For client-side attacks, a filtering proxy is the most effective; address space layout randomization and space execution prevention is less potent than on server-side attacks.

The scenarios estimated in this study did not specify all of the variables that could be relevant. The undefined variables (e.g., the type of service that is vulnerable) certainly vary among and within enterprises. As a result, it is impossible to say how much of the uncertainty arises from variations among unspecified variables in enterprises (i.e., aleatory uncertainty) and how much arises from the expert's lack of knowledge about arbitrary code attacks (i.e., epistemic uncertainty). However, it is reasonable to expect that both types of uncertainty contribute to the spread of the estimated intervals.

The variables included in this study were drawn from the literature with the assistance of domain experts with practical experience from arbitrary code execution attacks and the effectiveness of several of those variables was evaluated in a quantitative pilot study [11] The hypothesis was that these variables make up a good model for predicting the probability of successful remote arbitrary code execution. The respondents of the survey were asked to improve this model by replacing one of the variables with a new variable of their own choice. Three of the respondents suggested changes to the model. In terms of the calibration score, these three variables are ranked third, eighth and eighteenth. Two of those respondents (ranked third and eighth) suggested that the implementation of NX should be detailed in the model, e.g., if it is the implementation for Linux Red Hat 4.1 or Windows XP SP2. One respondent (ranked eighteenth) would like to replace ASLR with the existence of a host-based intrusion detection system in the targeted machine. The fact that only three of the 21 respondents suggested changes to the model indicates that it successfully captured the most important variables. However, future work in this field could add more detail to the scenario descriptions to identify the differences between different NX implementations and to investigate the impact of host-based intrusion detection.

7 Conclusions

The synthesized judgment of domain experts is that the most effective measure against server-side arbitrary code execution attacks is to implement access controls that limit the functionality that attackers can use. However, deep-packet inspection firewalls and measures available in operating systems (*ASLR* and *NX*) also lower the probability of successful compromise. For client-side attacks, where an application client is exposed to malicious data, the most effective countermeasure is the use of a filtering proxy. Operating system measures do not have as strong effects on attacks against clients. Decision-makers in enterprises should consider these effects when they contemplate measures against code injection attacks.

However, while these synthesized judgments provide valuable input to decision-makers and researchers, they come with a substantial amount of uncertainty. Further research could add more detailed variables to the attack scenarios to remove aleatory uncertainty. Also, this would enable more detailed data collection from experiments or observations to remove epistemic uncertainty. The results from this study can provide valuable information to future studies in this direction e.g., the approximate importance of the studied variables and that they are perceived to be fairly independent.

8 References

[1] Y. Younan, "Efficient countermeasures for software vulnerabilities due to memory management errors," Katholieke Universiteit Leuven, 2008.

[2] D. Patsos, S. Mitropoulos, and C. Douligeris, "Expanding topological vulnerability analysis to intrusion detection through the incident response intelligence system," *Information Management & Computer Security*, vol. 18, no. 4, pp. 291-309, 2010.

[3] T. Sommestad, M. Ekstedt, and P. Johnson, "A Probabilistic Relational Model for Security Risk Analysis," *Computers & Security*, 2010.

[4] R. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *13th European Symposium on Research in Computer Security (ESORICS)*, 2008, no. 0716665, pp. 18-34.

[5] J. Homer, K. Manhattan, X. Ou, and D. Schmidt, "A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks," Kansas, 2010.

[6] H. Shacham, M. Page, B. Pfaff, E. J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298–307.

[7] J. Wilander and M. Kamkar, "A comparison of publicly available tools for dynamic buffer overflow prevention," in *Proceedings of the 10th Network and Distributed System Security Symposium*, 2003, pp. 149–162.

[8] R. Cooke and L. Goossens, "Expert judgement elicitation for risk assessments of critical infrastructures," *Journal of Risk Research*, vol. 7, no. 643–656, 2004.

[9] J. J. C. H. Ryan, T. a. Mazzuchi, D. J. Ryan, J. Lopez de la Cruz, and R. Cooke, "Quantifying information security risks using expert judgment elicitation," *Computers & Operations Research*, pp. 1-11, Dec. 2010.

[10] R. Cooke, *Experts in Uncertainty: Opinions and Subjective Probability in Science*. New York, New York, USA: Open University Press, 1991.

[11] H. Holm, T. Sommestad, M. Ekstedt, and U. Franke, "Expert assessment on the probability of successful remote code execution attacks," in *Proceedings of 8th International Workshop on Security in Information Systems - WOSIS*, 2011.

[12] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole, "Buffer Overflows : Attacks and Defenses for the Vulnerability of the Decade," in *Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems]*, 2003, pp. 227-237.

[13] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in

Published by FIRST-Forum of Incident Response and Security Teams, 2007, pp. 1-23.

[14] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems," Gaithersburg, MD, USA, 2007.

[15] A. Fink, J. Kosecoff, M. Chassin, and R. H. Brook, "Consensus methods: characteristics and guidelines for use.," *American journal of public health*, vol. 74, no. 9, pp. 979-83, Sep. 1984.

[16] A. H. Ashton, "Does consensus imply accuracy in accounting studies of decision making?," *The Accounting Review*, vol. 60, no. 2, pp. 173–185, 1985.

[17] D. J. Weiss and J. Shanteau, "Empirical assessment of expertise," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 1, p. 104, 2003.

[18] M. J. Abdolmohammadi and J. Shanteau, "Personal attributes of expert auditors," *Organizational Behavior and Human Decision Processes*, vol. 53, no. 2, pp. 158–172, 1992.

[19] J. Shanteau, D. J. Weiss, R. P. Thomas, and J. C. Pounds, "Performance-based assessment of expertise: How to decide if someone is an expert or not," *European Journal of Operational Research*, vol. 136, no. 2, pp. 253–263, 2002.

[20] R. T. Clemen and R. L. Winkler, "Combining probability distributions from experts in risk analysis," *Risk Analysis*, vol. 19, no. 187, pp. 187-204, 1999.

[21] R. M. Cooke, *Experts in Uncertainty: Opinions and Subjective Probability in Science*. New York, New York, USA: Open University Press, 1991.

[22] R. M. Cooke, "TU Delft expert judgment data base," *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May 2008.

[23] U. S. D. of C. NIST Computer Security Resource Center (CSRC), "National Vulnerability Database," 2011. [Online]. Available: www.nvd.nist.org. [Accessed: 28-Apr-2011].

[24] F. Bolger and G. Wright, "Assessing the quality of expert judgment: Issues and analysis," *Decision Support Systems*, vol. 11, no. 1, pp. 1-24, Jan. 1994.

[25] Elsevier B.V., "Scopus," 2011. [Online]. Available: http://www.scopus.com/.

[26] Elsevier Inc, "Engineering Village," 2011. [Online]. Available: http://www.engineeringvillage.com. [Accessed: 24-Feb-2011].

[27] S. T. Cavusgil and L. A. Elvey-Kirk, "Mail survey response behavior: A conceptualization of motivating factors and an empirical study," *European Journal of Marketing*, vol. 32, no. 11/12, pp. 1165–1192, 1998.

[28] P. H. Garthwaite, J. B. Kadane, and A. O'Hagan, "Statistical methods for eliciting probability distributions," *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 680-701, 2005.

[29] L. J. Cronbach and R. J. Shavelson, "My Current Thoughts on Coefficient Alpha and Successor Procedures," *Educational and Psychological Measurement*, vol. 64, no. 3, pp. 391-418, Jun. 2004.

[30] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

Paper D: Quantifying the effectiveness of intrusion detection systems in operation through domain experts

Teodor Sommestad, Hannes Holm, Mathias Ekstedt and Nicholas Honeth

Abstract

An intrusion detection system is a security measure that can help system administrators in enterprise environments to detect attacks made against networks and their hosts. Evaluating the effectiveness of IDSs by experiments or observations is however difficult and costly. This paper describes the result of a study where 165 domain experts in the intrusion detection field estimated the effectiveness of 24 different scenarios pertaining to detection of remote arbitrary code exploits.

1 Introduction

Enterprise information systems are essential to most organizations. They store important information and facilitate both critical and supportive business processes. Enterprise information systems are also complex and comprise infrastructure, application software, business processes, and humans. These heterogeneous and dynamic environments are at constant risk from external and internal attacks. To secure these complex systems is difficult. Best practice in system security management involves application of a wide range of measures. Intrusion detection systems (IDS) are promising security measures that are commonly used to defend information systems [1]. An IDS monitors a computer network or its hosts to detect attacks made against them. Once attacks are identified, administrators can be notified and appropriate actions can be performed. Since IDSs can detect a wide range of attacks they may be used to monitor entire computer networks where they can complement other enterprise security measures on enterprise-level.

However, IDSs are not perfect. They fail to detect attacks that take place and raise alarms for events that are not actual attacks. In practice a detection rate must be traded off against the false alarm rate for the IDS. The effectiveness of an IDS can be determined after such a trade-off has been made. In this paper effectiveness is defined as in [2]: the probability that the administrator reacts appropriately when an attack occurs.

The development of models and techniques for IDSs dates back three decades [3], [4] and even though there exist a wide range of IDS solutions on the market today, IDSs are still a viable research field and much improvement is needed before they operate perfectly. How effective an enterprise's IDS is in different operating conditions is largely unknown. Several variables are believed to impact the effectiveness of an IDS in operation. For example, if the rules or models of IDS are updated, if the IDS has been tuned for its environment, and if it is host based or network based [5]. For a decision-maker who considers installing or adjusting an IDS, the impact of such variables are of high relevance to guide them in making effective system design decisions.

Quantitative studies have been made on some such aspects. For example, the impact of tuning configuration parameters in certain platforms [6], how the detection rate depends on its host's hardware performance [7], or how well different IDS products detect network scans [8]. Many qualitative evaluations (e.g., [9]) are also available. Investigating the effectiveness of IDSs and different IDS configurations in realistic environments is, however, difficult and costly. For instance, an extensive comparative study of 18 different IDSs was made during 1998 and 1999 by the Lincoln Laboratory at MIT [10], [11]. However, significant shortcomings have been identified in this study by [12], for example with the realism of the background data used during the tests. In general, several challenges have been identified for empirical tests of IDSs [13], [14]. As a consequence, few reliable empirical studies on the effectiveness of IDSs can be found. In fact, the only study which addresses their operational effectiveness is the experiment described in [15]. Although the experiment described in in [15] was associated with considerable cost it is limited to a large set of assumptions, e.g., concerning the attacks to be detected and how the IDS is managed. The absence of such guidance impedes effective use of this type of system in enterprises.

Expert judgment is often used when quantitative data is difficult to obtain from empirical studies or by other means. It has been used to assess the importance of attributes related to critical infrastructure risks [16], to quantify uncertainties related to crops [17] and recently to assess strategies related to security [18]. More examples of successful applications can be found in [19]. This paper describes a study in which a survey was used to collect expert judgment that quantifies the effectiveness of signature based IDSs in different operational scenarios. The experts in this study are researchers in the IDSs field who used their domain knowledge to assess whether arbitrary code execution attacks would be detected by an administrator. These assessments were made for 24 different operational scenarios. The respondents' judgments was synthesized with an established method that assigns weights to domain experts' judgment based on their performance on a number of test questions. The uncertainty of these estimates and the practical issues related to the implementation of different scenarios are also described.

The paper is structured as follows. Section two presents the operational scenarios that were investigated and the variables used to specify these. In section three Cooke's classical method for expert judgement is explained. This method is used to sort out experts that produce calibrated assessments. Section four presents the data collection method. Section five presents the results are estimates of IDS's effectiveness in 24 operational scenarios, and variables influence on this effectiveness. In section six these results are discussed and in section seven conclusions are drawn.

2 Operational scenarios – a prediction model for IDS effectiveness

The quality of IDSs can be evaluated by a number of criteria [20]. In accordance with Axelsson's [2] definition of effectiveness, this research investigates the probability that actual attacks are detected and reacted upon by the administrator monitoring the IDS. The attacks for which effectiveness is investigated in this study are the types of attack where arbitrary code is remotely executed on the targeted machine. A number of operational scenarios for IDSs were investigated. These operational scenarios were specified by selecting a number of variables based on a literature review and consultation with three security experts working in the field of IDSs. A summary of the variables identified in the literature review is presented in section 2.1; the variables used in the present study are described in and 2.2.

2.1 Literature review

A plethora of detection methods and techniques have been introduced since the work began on intrusion detection in the 1980's with work as [3] and [4]. A number of articles divide these into broad classification schemes. A common division is made between anomaly based intrusion detection and signature (or misuse) based intrusion detection [20–22]. Anomaly based intrusion detection estimates the normal behaviour of a system and generates an alarm when the deviation from the normal exceeds some threshold [22]. Signature based schemes look for patterns (signatures) in the analysed data and raise an alarm if the patterns match a known attacks [22]. Some classifications also distinguish specification based engines from these two schemes [9]; others regard specification-based engines as a subset to anomaly based detection [21]. In specification based engines activity that deviates from predefined constraints (e.g., descriptions of correct behaviour) would cause alarms. Hybrids or compound solutions are also possible [5], [21].

Anomaly based detection schemes have been given most of the attention in recent research on intrusion detection systems. [22] describes techniques used by these systems to detect anomalies such as: statistical based, knowledge based or machine learning based. [20] divide them into: statistical, sequence matching and learning, predictive pattern generation and neural networks.

Signature based detection schemes also come in different variants. [21] divides them into: state-modelling, expert system, string matching and simple rule based. [20] divides them into: expert systems, keystroke monitoring, model based, state transition analysis and pattern matching. While most research has been performed on anomaly based detection in recent years, most IDSs that are commercially available and used in practice are signature based [23].

The detection model can make a difference to the effectiveness of an IDS. It is often noted that signature based detection only detects attacks that correspond to known signatures while anomaly based detection also can detect previously unknown attack types (see for example [22]). The coverage, i.e., the attack types the IDS can detect, is of obvious relevance to the effectiveness in practice [12], [14]. The algorithm used also makes a difference. For instance, [24] compares a number of anomaly detection algorithm and show a clear difference in their performance. Much research effort has been dedicated to algorithms for detection. However, a wide range of other variables are also of importance to the effectiveness of operational IDSs.

Whether sensors are placed on network hosts, in the network infrastructure, or on both, make a difference [5]. Another variation with a potential impact is if sensors are placed so that they listen to the network passively (e.g., through a spanning port or network tap) or if it is placed inline so that all traffic must go through it [5], [25]. Host based sensors could also be placed inside or outside the code they are supposed to monitor and as this influences what the sensor can monitor it will also have an impact on the effectiveness [25]. The protocols, protocol layers and the amount of traffic which the IDS can handle are also of relevance [5], [14]. Furthermore, the environment which the sensors are placed in can be expected to influence the effectiveness [12], [14]. Complex and intensive network traffic may for example give rise to higher instances of false alarms and make it difficult for the IDS to identify actual attacks.

There are, in addition to detection mechanism, the placements of sensors and the environment, a number of variables related to deployment and management of the IDS that can be expected to influence its performance. Configuration and tuning is of importance to both anomaly based and signature based intrusion detection [5]. Configuration parameters include: thresholds and alert settings to optimize false positives and false negatives [5], [26], tuning and customizing the system for its environment [5], [26] and securing the actual IDS from attacks [5], [14]. Deploying an IDS correctly is generally challenging and as a consequence the competence of system administrators is an important factor [5], [26]. For example, the administrators' programming skills and their knowledge about the environment where they are supposed to deploy the IDS on are of relevance [5], [26].

After deployment the detection system needs to be maintained and managed. Updating the system and its engine to the latest version is part in this management process [5]. For signature based detection system it is of the essence to maintain the signature database updated [5]. Periodic testing of the IDS's functionality has also been suggested [5].

Alarm lists may comprise of as much as 99 % false alarms and methods that assist the administrator in identifying actual attacks are therefore important [27]. The abovementioned variables influence the amount of false alarms that are raised by the IDS and the amount of attacks it misses. In the end, however, an administrator must be able to distinguish actual attacks from false positives and decide what to react upon. It is the effectiveness achieved in this stage that is investigated in this paper. In [28] design recommendations have been put forward to ease the cognitive burden placed on administrators using visualization. Research in this field (e.g., [29], [30]) has presented different techniques for visualization. While visualization of alarms and the network's status can help the administrator, the competence of this person is also an important factor. [31] have found that administrators require expertise in networking, security, and a portion of situated expertise (e.g., about the enterprise to work in) to solve their task. Moreover, they are often faced with problems that are not predefined and change as the environments evolve [31].

2.2 Variables specified in the assessed scenarios

As described in section 2.1 there are numerous variables that may influence the effectiveness of an IDS in operation. One could specify operational scenarios by assigning values to all of these variables, e.g., regarding the employed algorithm(s), the competence of the operators and the profile of the background traffic. However, doing so would only show the value of these exact configurations and limit the validity of the result to these particular cases. Collecting such detailed information in an enterprise-context would also be extremely expensive and prediction models requiring this level of detail would be expensive to use. Also, as shown by the critique against experimental efforts [12], it is difficult to identify all variables that are of relevance and make sure that they are representative for typical operations.

This study aims to provide approximate values for IDSs' effectiveness and the approximate importance of a number of important variables related to them. To maintain generality of the estimates produced it focuses on a number of carefully selected variables and let the greater majority of variables vary as they typically do in an enterprise environment. The variation between enterprises of variables that influence effectiveness (e.g., how competent administrators are) will make the exact effectiveness uncertain as it will vary from enterprise to enterprise. This uncertainty is managed by expressing the effectiveness through a probability distribution that captures the uncertainty caused by this noise. Hence, for each operational scenario the experts were asked to provide estimates of effectiveness in terms of a probability distribution that was representative for enterprises, given that unspecified variables vary as they do in practice.

The selection of variables to include in the operational scenarios was made by consulting three experts on IDSs. These domain experts were presented with a list of variables and were asked to complement this list with other variables they found important. They were then asked to prioritize these variables based on their utility for making predictions on the effectiveness of an intrusion detection solution used in practice. As the selected variables were to be used for predictions the respondents were asked to not only consider their impact on the effectiveness, but also the system owner's possibility to identify their values for an installation. The respondents were also asked to identify meaningful assumptions which would have a limited effect on the usability of the result. That is, assumptions about conditions that apply to most enterprises or for other reasons correspond to scenarios of interest to decision makers in enterprises. Based on this prioritization procedure two conditions were assumed and five variables were selected, forming the model depicted in Figure 1.

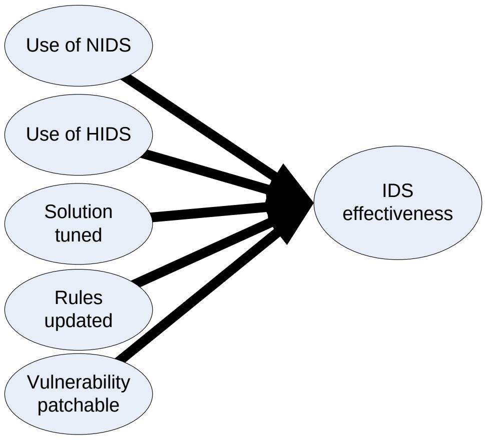


Figure 1. Variables studied.

Assumptions were made on the conditions of the attack scenarios they are exposed to and the detection scheme used by the IDS. The attack scenario was specified as remote arbitrary code exploit performed by a professional penetration tester with the possibility to spend one week on preparing the attack. Thus, the attacker exploits a software vulnerability in order to execute code on the targeted system. The professional penetration tester should be assumed to be an outsider. The detection scheme was assumed to signature based. The domain experts experience was that the vast majority of IDSs installed in enterprises today use this detection scheme as it is more mature. They were therefore considered more interesting for decision makers to assess. Table 1 describes the five variables that were used to describe the different operational scenarios. In total 24 different operational scenarios are investigated, each corresponding to a specific configuration of the five variables.

Two of the five variables concern the placement of sensors – if the IDS is host or network based. Another variable selected by the domain experts was the tuning of the IDSs. Whether they were tuned for their environments or not was therefore included as one variable. Updates of signatures used by the detections system was regarded as an important variable. Finally, the type of vulnerability that was to be exploited was judged to be important. A signature based IDS is presumably less effective in scenarios with unknown attacks, as noted in section 2.1. In this model the vulnerability-type exploited is captured by considering scenarios where the attack is possible to patch with a software update (i.e., is well-known vulnerabilities) as well as scenarios where the exploit uses software vulnerabilities for which no patch is available. This variable was more highly prioritized than the exact signature match to exploits used by the attacker due to the fact that details on the latter are difficult to collect in practice.

All possible combinations of these five variables are considered except for those where there is neither a network based IDS or a host based IDS. This resulted in the 24 scenarios described in Table 3. Each of these corresponds to a specific state in variables and is associated with a probability distribution for effectiveness.

Table 1. Variables included in the model.

Variable	Description
NIDS	Whether a network based intrusion detection system is used or not.
HIDS	Whether a host based intrusion detection system is used or not.
Tuned	Whether the intrusion detection systems used have been tuned for their environment or not.
Updated	Whether the signatures used by the intrusion detection systems are fully updated or not.
Patchable	Whether the exploit they are supposed to detect use a vulnerability that can be patched or not.

3 Method used to synthesize expert judgments

This paper uses the judgment of domain experts to produce quantitative estimates of IDSs' effectiveness in different scenarios. There is a substantial amount of research on how to combine, or synthesize, the judgment of multiple experts to increase the calibration of the estimates used. These techniques include the following: consensus methods [32], [33], the Cochran-Weiss-Shanteau index [34], self-proclaimed expertise

[35], experience [36], certifications [36], peer-recommendations [36], and Cooke's classical method [37]. There is little research that compares the accuracy that these methods yield. However, research has shown that groups of individuals assess an uncertain quantity better than the average expert, while the best individuals in the group are often better calibrated than the group as a whole [38].

The scheme used to combine judgments in this research is the one used in the classical model of Cooke [37]. Cooke's model is a generic method for combining expert judgments that has been applied to a number of different domains. Experience from applications of Cooke's classical method has shown that it outperforms both the best expert and the "equal weight" combination of estimates. In an evaluation involving 45 studies, it performed significantly better than both alternatives in 27 studies and equally well as the best expert in 15 of the studies [19].

In Cooke's classical method two scores, one for *calibration* and one for *information*, are calculated for the respondents for the purpose of weighting them. The scores are based on the respondents' answers to a set of seed questions, i.e., questions for which the true answer is known at the time of analysis. The calibration score shows how correctly a respondent's answers reflect the true value and the information score shows how precise a respondent's answer is. These two scores are used to assign weights to the respondents based on their performance, via a function called a virtual *decision maker*. The weights defined by this decision maker are then used to weight the respondents' answers to the questions of interest – in this case the operational scenarios described in section 5.1.

In summary, the method thus filters out individuals as "true experts" from a pool of potential experts, given their accuracy and preciseness in the answers of a set of test questions. For the questions of interest then, only those filtered out as "true experts" are used. Thus, this means that the vast amount of initial respondents' answers is simply disregarded. This means that, in contrast to many other expert based studies, the fundamental philosophy of this method is that it does not really matter how many respondents that participates in the study as long as at least one can be considered a "true expert" (according to the method), whose answers could be trusted as "true values". In sections 3.1, 3.2 and 3.3 Cooke's classical method is explained. For a more detailed explanation the reader is referred to [37].

3.1 Calibration score

In the elicitation phase the experts provide individual answers to the seed questions. The seed questions request the respondents to specify a probability distribution for an uncertain continuous variable. This distribution is typically specified by stating its 5th, 50th, and 95th percentile values. This yields four intervals over the percentiles *[0-5, 5-50, 50-95, 95-100]* with probabilities of *p= [0.05, 0.45, 0.45, 0.05]*. As the seeds are realizations of these variables, the well calibrated expert will have approximately 5% of the realizations in the first interval, 45 % of the realizations in the second interval, 45 % of the realizations in the third interval and 5% of the realizations in the fourth interval. If *s* is the distribution of the seed over the intervals, the relative information of *s* with respect to *p* is: $I(s,p) = \sum_{i=1}^{4} \ln(s_i/p_i)$. This value indicates how surprised someone would be if one believed that the distribution was p and then learnt that it was *s*.

If *N* is the number of samples/seeds the statistic of $2NI(s, p)$ is asymptotically Chi-square distributed with three degrees of freedom. This asymptotic behaviour is used to calculate the calibration *Cal* of expert *e* as: $Cal(e) = 1 - \chi_3^2(2N\, I(s,p))$. Calibration measures the statistical likelihood of a hypothesis. The hypothesis tested is that realizations of the seeds (*s*) are sampled independently from distributions agreeing with the expert's assessments (*p*).

3.2 Information score

The second score used to weight experts is the information score, i.e., how precise and informative the expert's distributions are. This score is calculated as the deviation of the expert's distribution to some meaningful background measure. In this study the background measure is a uniform distribution over [0,1].

If b_i is the background density for seed $i \in \{1,\ldots,N\}$ and $d_{e,i}$ is the density of expert *e* on seed *i* the information score for expert *e* is calculated as: $\inf(e) = \frac{1}{N}\sum_{i=1}^{N} I(d_{e,i}, b_i)$, i.e., as the relative information of the experts distribution with respect to the background measure. It should be noted that the information score does not reflect calibration and does not depend on the realization of the seed questions. So, regardless of what the correct answer is to a seed question, a respondent will receive a low information score for an answer which is similar to the background measure, i.e., the answer is distributed evenly over the variable's range. Conversely, an answer which is more certain and assigns most of the probability density to a few values will yield a high information score.

3.3 Constructing a decision maker

Cooke's classical method rewards experts who produce answers with high calibration (high statistical likelihood) and high information value (low entropy). A strictly proper scoring rule is used to calculate the weights the decision maker should use. If the calibration score of the expert *e* is at least as high as a threshold value (α) the expert's weight is obtained by $w(e) = Cal(e) * Inf(e)$. If the experts calibration is less than the threshold value (α) the expert's weight is set to zero, a situation which commonly happens to a substantial portion of experts in practical applications.

The threshold value α corresponds to the significance level for rejection of the hypothesis that the expert is well calibrated. The value of α is identified by resolving the value that would optimize a virtual decision maker. This virtual decision maker combines the experts' answers (probability distributions) based on the weights obtained at the chosen threshold value (α). The optimal level for α is where this virtual expert would receive the highest possible weight if it was added to the expert pool and had its calibration and information scored as the actual experts.

When α has been resolved, the normalized value of the experts' weights *w(e)* are used to combine their estimates of the uncertain quantities of interest.

4 Data collection method

This section presents how the data was collected by explaining: which population and sample of experts that was chosen, how the measurement instrument was developed and tested, how seed questions for Cooke's classical method were assessed, and the result of applying Cooke's classical method.

4.1 The domain experts

As this research aims to identify quantities related to IDSs the respondents needed both the ability to evaluate aspects in the domain and the ability to reason in terms of probabilities. In terms of the expert categories described in [34] individuals that are expert judges are desirable. Studies of experts' calibration have concluded that experts are well calibrated in situations with learnability and with ecological validity [39]. Learnability comes with models over the domain, the possibility to express judgment in a coherent quantifiable manner and the opportunity to learn to from historic predictions and outcomes. Ecological validity is present if the expert is used to making judgments of the type they are asked in the survey.

Respondents that have had the opportunity to learn the effectiveness of IDSs are likely to be those that have performed tests on different solutions in a quantifiable manner. Researchers in the intrusion detection field have performed and disseminated a number of empirical studies related to effectiveness of different solutions. While these studies are questionable with respect to generality [12] they do offer input to specific scenarios. Practitioners (e.g. system administrators) will probably not have the same opportunity to learn the effect of different scenarios since they typically only have experience from a few installations and rarely perform stringent evaluations of effectiveness. Also, with respect to ecological validity it is expected that researchers are more used to estimating probability distribution and reason in terms of probabilities.

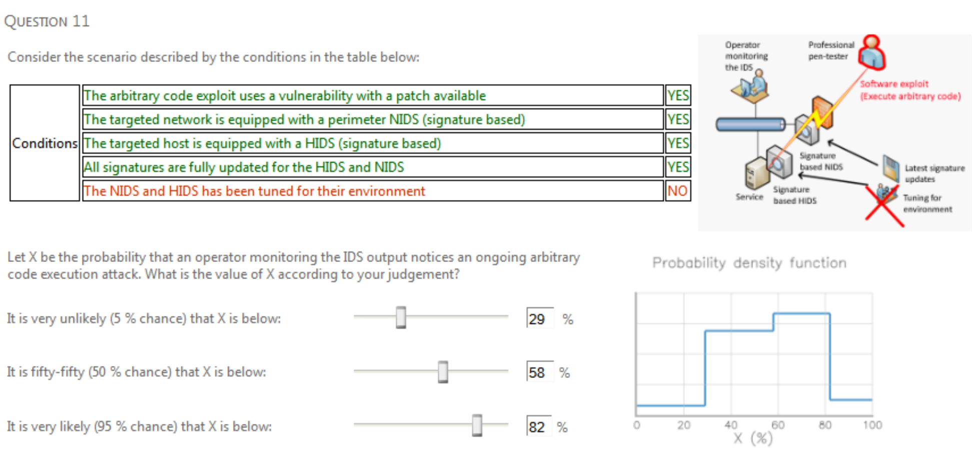
For these reasons IDSs researchers were chosen as the respondents to the survey. To identify suitable respondents, articles published in the SCOPUS database [40] between January 2005 and September 2010 were reviewed. Authors who had written articles in the information technology field with "intrusion detection" in the title, abstract or keywords were identified. If their contact information could be found they were added to the list of potential respondents, resulting in a sample of 13561 respondents. After reviewing respondents with respect to their research topic, and the availability of their contact information, a sample of 6269 individuals was identified. Of these, the contact information to at approximately 1550 turned out to be incorrect or out-dated. A pilot study involving 500 respondents (described in section 4.2) reduced the number of respondents who received the final survey to approximately 4200 individuals.

Out of approximately 4200 researchers invited to the survey 1355 opened it and 243 submitted answers to the survey's questions. A response rate of this magnitude is to be expected of a slightly more advanced survey. As recommended by [41], motivators were presented to the respondents invited to the survey: (i) helping the research community as whole, (ii) the possibility to win a gift certificate on literature, and (iii) being able to compare their answers to other experts after the survey was completed. A number of respondents provided input on less than half of the questions, i.e., they answered with the pre-set background measure on more than half of the questions. These were excluded from further analysis, resulting in 165 usable surveys completed by IDS researchers.

4.2 Elicitation instrument

A web survey was used to collect the probability distributions from the invited respondents. The survey comprised four parts, each beginning with a short introduction to the section. First, the respondents were given an introduction to the survey that explained the purpose of the survey and its outline. In this introduction they also confirmed that they were the person who had been invited and provided information about themselves, e.g. years of experience in the field of research. Second, the respondents received training regarding the answering format used in the survey. After confirming that this format was understood the respondents proceeded to its third part. In the third part both the seed questions and the questions of the study were presented to the respondents. Finally, the respondents were asked to provide qualitative feedback on the survey and the variables covered by it.

The questions in section three of the survey were each described through a scenario entailing a number of conditions. Scenarios and conditions for the seed questions can be found in Table 2; scenarios and conditions for the questions at issue in this study can be found described in Table 3. For each scenario the respondent was asked to provide a probability distribution that expressed the respondent's belief. As is custom in applications of Cooke's classical method this probability distribution was specified by setting the 5th percentile, the 50th percentile (the median), and the 95th percentile for the probability distribution. In the survey the respondents specified their distribution by adjusting sliders or entering values to draw a dynamically updated graph over their probability distribution. The three points specified by the respondents defines four intervals over the range [0, 100]. The graphs displayed the probability density as a histogram, instantly updated upon change of the input values. Use of graphical formats is known to improve the accuracy of elicitation [42]. Figures and colours were also used to complement the textual questions and make the questions easier to understand. In Figure 2 the format presented to the respondents is exemplified. A generic figure describing the survey's variables could also be found at the top of each section, along with introductory text.



QUESTION 11

Consider the scenario described by the conditions in the table below:

Conditions	The arbitrary code exploit uses a vulnerability with a patch available	YES
	The targeted network is equipped with a perimeter NIDS (signature based)	YES
	The targeted host is equipped with a HIDS (signature based)	YES
	All signatures are fully updated for the HIDS and NIDS	YES
	The NIDS and HIDS has been tuned for their environment	NO

Let X be the probability that an operator monitoring the IDS output notices an ongoing arbitrary code execution attack. What is the value of X according to your judgement?

It is very unlikely (5 % chance) that X is below: 29 %

It is fifty-fifty (50 % chance) that X is below: 58 %

It is very likely (95 % chance) that X is below: 82 %

Figure 2. Example of question and answering format in the survey.

Elicitation of probability distributions is associated with a number of issues [42]. Effort was therefore spent on ensuring that the measurement instrument held sufficient quality. The survey was, after careful construction, qualitatively reviewed during personal sessions with two external respondents representative of the population. These sessions contained two parts. First the respondents were given task to fill in the survey, given the same amount of information as someone doing it remotely. After this discussions followed regarding the instrument quality. These sessions resulted in several improvements with respect to language and phrasing of questions.

The main part of the instrument review however took place in the next phase: a pilot study using a randomized sample of 500 respondents from the previously mentioned 6269 screened subjects. This pilot survey was opened by 123 persons and completed by 34 during the week it was open. Cronbach's alpha [43], [44] is often used to test the reliability of a survey instrument and if respondents understand its questions. A reliability test using Cronbach's alpha was carried out using one variable (four different versions of the fourth seed question). Measuring the reliability of more than one question would be inefficient, as all sections and questions were formatted in the same way, and most likely have created bias for the instrument used during the pilot study. Results from this test showed a reliability value of 0.817, which indicates good internal consistency of the instrument. Qualitative comments also confirmed that respondents understood the questions. A few possible improvements were however identified. After these changes had been implemented the survey was again qualitatively reviewed by the two persons mentioned previously.

4.3 Seed questions

In this study Cooke's classical method is used to synthesize experts' judgements. This method assigns weight to the experts based on their calibration and information score to the seed questions. As an expert's performance on answering the seed questions is used to weight them, it is critical that the seeds are highly validated and that they lie in the same domain as the studied variables. Thus, the seeds should represent the truth and it should be difficult to tell them apart from the questions in the study. However, they do not necessarily need to be directly related to questions of the study [45].

Naturally, the robustness of the weights attributed to individual experts depends on the number of seeds used. Experience shows that around eight seed questions are enough to see substantial difference in calibration [45].

For this study two types of seed questions were used (cf. Table 2). The first type (questions 1-3) concerned the detection rate of different IDS products when faced with a seven types of commands produced with Nmap [46], a network discovery tool. The actual detection rates (the realization values) were drawn from an empirical test described in [8]. The second type of seed questions (4-8) concerned the coverage of software vulnerabilities in the IDS ruleset maintained by the Sourcefire Vulnerability Research Team [47]. This ruleset is used in the popular signature based IDS product Snort [48], amongst others. Statistics on how well this ruleset covered vulnerabilities in different products and timeframes was obtained by cross referencing this ruleset's coverage to the National Vulnerability Database [49]. The Common Vulnerability Scoring System

(CVSS) [50] is a well-established system for rating a software vulnerability's severity. Vulnerabilities rated with high severity according in the Common Vulnerability Scoring System (CVSS) [50] were used as such vulnerabilities are those that could be used for arbitrary code exploits.

Table 2. Seed questions used in abbreviated format. The seven NMAP commands can be found in [8]**.**

#	Question	Realization (%)
1	If one of the seven NMAP commands was randomly selected and then executed, how probable do you think it is that a default configured Snort intrusion detection system would detect it?	72
2	If one of the seven NMAP commands was randomly selected and then executed, how probable do you think it is that a default configured Tamandua intrusion detection system would detect it?	29
3	If one of the seven NMAP commands was randomly selected and then executed, how probable do you think it is that a default configured Firestorm intrusion detection system would detect it?	29
4	Consider vulnerabilities of high severity (according to CVSS) that impacts Windows 7 and was published during 2010. What portion of these vulnerabilities has a corresponding signature in Snort's default ruleset?	40
5	Consider vulnerabilities of high severity (according to CVSS) that impacts MySQL and was published during 2004-2009. What portion of these vulnerabilities has a corresponding signature in Snort's default ruleset?	87
6	Consider vulnerabilities of high severity (according to CVSS) that impacts Windows 7 and was published during 2009. What portion of these vulnerabilities has a corresponding signature in Snort's default ruleset?	37
7	Consider vulnerabilities of high severity (according to CVSS) that impacts Windows 7 and was published during the last 6 months. What portion of these vulnerabilities has a corresponding signature in Snort's default ruleset?	35
8	Consider vulnerabilities of high severity (according to CVSS) that impacts Samba and was published during 2010. What portion of these vulnerabilities has a corresponding signature in Snort's default ruleset?	33

A threat to the validity of the results is that these sources are also available to the respondents who could have used them identify the answers to the seed questions. However, it appears unlikely that any of the respondents had done so. None of the respondents answering the survey have given comments that indicate that they have realized that the correct answer can be found this way. The qualitative reviewers did not realize this during the dry runs either. Furthermore, inspections of the answers received do not indicate any answers based on these sources. Naturally, the authors of the article used [8] were excluded from the list of potential respondents.

4.4 Respondents' performance

For each respondent the weight was calculated from their answers to the seed questions. All 165 respondents completed the survey in less than one hour. As in many other studies involving expert judgment many of the experts were poorly calibrated. Their calibration score varied between $2.200*10^{-10}$ and 0.6638 with a mean of 0.1575; their information score varied between $8.620*10^{-7}$ and 3.293 with a mean of 0.8630. Figure 3 shows the information score and calibration score of the respondents (c.f. section 3 for an explanation of these values).

Cooke's classical method aims is to identify those respondents whose judgment is well calibrated and informative. The virtual decision maker was optimized at a significance level (α) of 0.6638. Consequently, the twelve rightmost respondents in Figure 3 received a weight higher than zero and the other 153 respondents received a weight of zero. As noted above it is not uncommon that a substantial number of respondents receive the weight zero with this method.

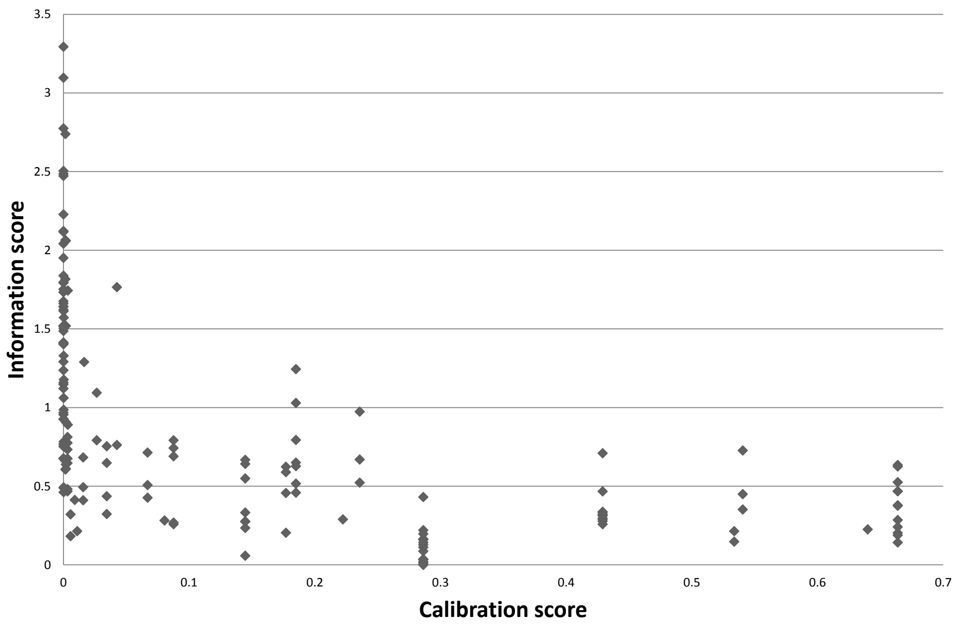


Figure 3. Information and calibration scores of the respondents

The twelve respondents who received a positive weight all had the same calibration score (0.6638). Their weights are therefore directly proportional with their information score (cf. section 3.3). They received weights between 0.0313 and 0.1401 after normalization.

5 Results

This section presents the result of the analysis performed on the judgment of the 165 researchers. In section 5.1 the synthesized estimates of those respondents who were assigned weight are presented. In section 5.2 the influence that each of the five individual variable have on the effectiveness is described.

5.1 Detection rate in the scenarios

To identify the probability distribution which the virtual decision maker assigns to the effectiveness in the 24 scenarios the individual estimates were combined using their weights. The estimated distributions were assumed to be distributed in the same way as they were presented to the respondents (c.f. section

4.2), i.e., as depicted in the histograms over the four ranges they constructed with their answers.

As depicted in Table 3 the synthesized estimates show clear differences among the scenarios. The median for the scenarios varies between 32% and 65%; the value at the 5th percentile varies between 2% and 13%; the value at the 95th percentile varies between 80% and 97%. Scenario one (where all variables are true) has the highest median (65%) and mean (58%) effectiveness. Scenario 17, which is the same as scenario 1 but without the network IDS, is the second most effective judging from the median (63%) and mean (55%). Scenario 4, 6, 8, 12, 14, 16 and 24 are on the other end of the scale with medians or means of 40 % or below.

Table 3. The scenarios, their variable configuration and their estimated effectiveness.

Scenario	NIDS	HIDS	Patchable	Updated	Tuned	Low (5%)	Median 50%)	High (95%)	Expected
1	Y	Y	Y	Y	Y	13	65	91	58
2	Y	Y	Y	Y	N	8	43	93	48
3	Y	Y	Y	N	Y	12	59	96	54
4	Y	Y	Y	N	N	5	39	82	41
5	Y	N	Y	Y	Y	6	48	91	47
6	Y	N	Y	Y	N	6	38	91	41
7	Y	N	Y	N	Y	8	44	88	44
8	Y	N	Y	N	N	4	32	92	39
9	Y	Y	N	Y	Y	9	51	91	48
10	Y	Y	N	Y	N	8	45	89	43
11	Y	Y	N	N	Y	10	49	90	46
12	Y	Y	N	N	N	2	39	80	38
13	Y	N	N	Y	Y	2	40	90	41
14	Y	N	N	Y	N	7	37	85	38
15	Y	N	N	N	Y	10	42	88	42
16	Y	N	N	N	N	2	39	93	43
17	N	Y	Y	Y	Y	8	63	94	55
18	N	Y	Y	Y	N	7	51	91	50
19	N	Y	Y	N	Y	9	53	92	50
20	N	Y	Y	N	N	4	48	97	47
21	N	Y	N	Y	Y	8	50	89	45
22	N	Y	N	Y	N	7	48	87	44
23	N	Y	N	N	Y	9	51	92	48
24	N	Y	N	N	N	2	40	84	40

5.2 Variables' influence on the effectiveness of intrusion detection

This study varies five variables in the scenarios. Both literature and domain experts have identified these variables as relevant to the effectiveness of an intrusion detection solution. The variation over scenarios on effectiveness supports this hypothesis. A relevant question is then how important these variables are for the IDS's effectiveness and if certain variable combinations have a particular effect, i.e., if the variables are independent or interact. Table 4 shows the mean influence that the five variables have on the probability distribution. It also shows the variable interactions with highest influence on the effectiveness.

The values in Table 4 show the weight of this variable or variable combination calculated as in a full factorial experiment [51]. These calculations are made under the assumption that scenarios without HIDS and NIDS will have zero effectiveness. The values thus represent the mean influence a variable or variable combination has on the effectiveness. For instance, the values for NIDS are obtained as: $\frac{1}{16}\sum_{i=1}^{16} Scenario_i - Scenario_{i+16}$, where scenario 25-32 have values zero (there is no detection system in place).

As can be seen from Table 4 the variables with highest influence are the NIDS and HIDS, i.e., to actually have an IDS. A NIDS do on average increase the expected effectiveness with 20.75 percentiles while a HIDS increase the expected effectiveness by 26.25 percentiles. The relatively high influence of these variables should be seen in the light that without them the effectiveness is zero. Given that a NIDS and/or HIDS is in place the most rewarding change is to tune the intrusion detection solution to its environment (4.125 % units). If the vulnerability that is exploited is patchable the expected effectiveness by 3.625 % units and if the IDS has the latest signatures influence by 1.625 % units.

Table 4. The influence strength of individual variables and selected variable combinations.

	Low (5%)	Median (50%)	High (95%)	Expected value
NIDS	3.6	19.1	44.0	20.8
HIDS	4.8	29.6	45.0	26.3
Tuned	2.7	7.3	1.8	4.1
Updated	0.8	2.8	0.5	1.7
Patchable	0.9	3.3	2.5	3.6
NIDS & HIDS	-2.0	-20.9	-45.8	-21.1
NIDS & Tuned	0.9	3.5	0.8	2.0
HIDS & Updated	1.1	2.0	1.0	1.8
HIDS & Patchable	0.5	2.8	1.8	2.8
Patchable & Updated	0.0	1.4	0.0	1.4

As can be seen from Table 4 the combination of a NIDS and HIDS has a substantial negative impact on the effectiveness. The negative value from this interaction is comparative to the individual influence they have. The interaction even exceeds the positive influence a NIDS have on the expected effectiveness. That is, having both a NIDS and a HIDS is on average less effective than having a HIDS only. Looking at the scenarios in Table 3, the negative numbers can be explained by the comparison of scenarios where no tuning has been made to the solution, i.e., if an un-tuned NIDS is removed and only an un-tuned HIDS is used the effectiveness increases. The negative value resulting from this interaction also exceeds the positive value a HIDS have on the 95th percentiles. The explanation for this negative influence can also be found in conjunction to un-tuned solutions. When the solution is neither updated nor tuned (as in scenario 4 and 12) the 95th percentile's value increases if the host based component is removed, given that a NIDS is in place.

Other variables also interact, but to a lesser extent in absolute numbers. Table 4 shows those interactions with influences greater than 1.25 % units (positive or negative) on the expected effectiveness. As can be seen, tuning appears to be of particular importance in the case where a NIDS is used. The expected value on effectiveness then increases by two percentiles in addition the 4.125 percentiles that tuning otherwise add, i.e., tuning is about 50 % more valuable if a NIDS is used. For HIDS, signatures that are updated increase the expected effectiveness by an extra 1.75 percentiles and HIDS also appears to be more helped by a scenario where the exploited vulnerability is possible to patch (i.e., is well known). The interaction is 2.75 percentiles between updates and vulnerability-type. The positive interaction between updating a system and being attacked with known (patchable) exploits is intuitive as updates can be expected to have a limited impact on the effectiveness against new attacks (which there seldom is a patch for).

6 Discussion

This outline of the discussion is as follows: Section 6.1 discusses the validity and reliability of the survey the experts' judgments and the survey as a knowledge elicitation instrument. Section 6.2 gives recommendations based on the research findings to practitioners and section 6.3 gives recommendations for future research.

6.1 Validity and reliability

Cooke's classical method [45] was used to synthesize expert judgments in this study. This performance based method aims to select the experts that are well calibrated and combine their judgments in an optimal way. The track record of this method (Cooke, 2008) positions it as a best-practice when it comes to eliciting expert judgment of uncertain quantities.

The answers on the seed questions show that many experts in the intrusion detection field are poorly calibrated (as in many other domains), i.e., their estimates do not match empirical observations well. This can be seen through the calibration scores to the seed questions used in this study (c.f. Table 2) and show to the importance of assigning different weights to experts' judgment. Twelve respondents were assigned weight when the virtual decision maker was optimized. As can be seen from Table 3 are the estimates from the twelve respondents who obtained weight provided relatively uninformative when compared to the respondents' estimates overall. This is should not be seen as surprising. Overconfidence is a well-known cause for poor calibration in expert judgments [52].

When using this method it is appropriate to perform robustness test with respect to the seed variables and the experts by removing one expert and investigating the impact of this removal [45]. Such tests were performed and indicate that the solution is robust to changes in both seed questions and experts.

Cooke [45] provides a list of guidelines for how to elicit data from experts: 1) questions must be clear and unambiguous, 2) a dry run should be carried out before the actual study, 3) an attractive graphical format should be used and there should be a brief explanation of the elicitation format, 4) elicitation should not exceed one hour , 5) coaching should be avoided and 6) an analyst should be present when respondents answer the questions. As described in section 4 all guidelines but 6) are met in this study, i.e., no analyst were present when respondents answered the questions. With a web survey this was obviously not fulfilled. The respondents were given contact information to the research group when invited to the survey that they were encouraged to use any if questions arose. While this ensures that no coaching occurred during the elicitation it is possible that it suppressed potential questions being asked. To identify potential issues of this type the respondents were asked to comment the clarity of the questions and the question format used. Based on the comment received no distressing issues relating to the questions formulations arose. Several respondents did however comment the difficulty of expressing knowledge quantitatively or the difficulty to estimate the effectiveness of IDSs in general (as there little empirical data on it). However, this issue is not surprising and is a part of the reason why this study was carried out in the first place.

The cost of obtaining observational data on the effectiveness of operational IDSs (were administrators use the system) was the main motivation for the use of IDS experts judgment to cover the broad scope of this study. The only observational data on this found in the literature is the one described in [15]. Although extensive efforts were made to arrange the experiment described in [15] (e.g., construction of fictive networks, installation and tuning of an IDS, time spent by attackers and administrators) it is associated with several assumptions and delimitations which threatens the representativeness of the result. It roughly corresponds to scenario 1 which the experts in this study assessed, i.e., the most ideal scenario. The experiment gave an effectiveness of 58% and the mean value of the domain experts is 59% (cf. Table 3). Thus, the experiment executed by [15] after this expert survey corroborates the experts' assessment.

6.2 Recommendations to information system decision-makers

From a practitioners' point of view these results provide input on which actions to take in order to achieve effective surveillance by an IDS. The results show that experts are uncertain about IDS effectiveness, and that many experts are poorly calibrated (incorrect and uncertain) on the test questions used to weight them. In other words, if a decision maker would ask a randomly selected IDS expert for advice (s)he is likely to get vague or incorrect suggestions; if multiple experts are asked for advice their recommendations will probably differ. This study has synthesized the judgment of a large number of security experts (out of whom the most calibrated have carefully been selected). The synthesized results are uncertain, but it is unlikely that the decision maker can get more precise knowledge (at this level of abstraction) from a random security expert or a random set of security experts. Also, knowing the uncertainty of the effectiveness in an IDS scenario will help the decision maker to make informed decisions and appreciate the effectiveness of countermeasures not covered by this study.

To tune the IDS to its environment is expected to increase the detection rate. However, tuning an IDS in an enterprise context is a continuous process: as soon as there has been a change in any parameter that is under surveillance the IDS needs to be tuned to reflect this change. For example, if the organization has installed a new FTP server or bought new computer systems the traffic patterns will change and the IDS will need to be tuned again. Since tuning requires constant adaptation of the IDS it will require that such system administrators regularly spend time analysing recent changes to the enterprise system architecture and adapt the IDS accordingly [5], [26]. Of course, these costs can be neglected if the IDS is deployed is static and documented environment, e.g., in an industrial facility's control system network.

If the IDS use the most recent ruleset its effectiveness will also increase. In comparison to tuning, keeping the IDS updated with a recent ruleset is a straightforward process which does not require administrators to analyse the current architecture or spend significant efforts on programming the IDS solution. On the other hand, subscriptions to new rules are often associated with some cost.

Host based solutions (HIDS) give a better effectiveness than network based solution (NIDS). However, a delimitation of HIDS's is that they are required to be implemented on a host-level, which could involve significant costs. For example, each HIDS might have to be manually installed on each supervised system, and perhaps manually tuned for the context of each such system. A NIDS-solution is not as effective as a HIDS-solution. As such, a cost effective architecture is likely to use a HIDS solution on the most sensitive systems in the enterprise and a NIDS solution to monitor less sensitive systems. For instance, a HIDS solution could be used to monitor critical business servers and a NIDS solution could be used to monitor office clients.

Combined solutions (with both HIDS and NIDS) are presented in literature and have the potential to increase the effectiveness. However, the result from this study suggests the opposite – a combination of a HIDS and NIDS is not believed to increase the effectiveness of intrusion detection. In fact, if a HIDS is already used, the experts believe that the effectiveness will decrease if an NIDS also is installed. One reason behind this could be that the output and the HIDS will overlap and that large amounts of information (and false alarms) that need to be parsed by the administrator in order to detect attacks with multiple sensors.

An interesting result of this study is that the possibility to patch the exploited vulnerability has the lowest impact of the assessed variables. This suggests that a signature based systems can detect novel attack-types, just as anomaly based systems.

Finally, organizational decision makers should reflect on whether IDSs really are needed in their environments. This study show that such tools are believed to only provide modest effectiveness, and to implement and maintain an IDS solution can be costly. The tools do not only require technical costs (installation/maintenance), but also the time of network administrators who need to carefully study the output of the solution to be able to detect real attacks.

6.3 Recommendations to researchers

Observational studies and experiments like the one described in [15] are costly to perform and should therefore be carefully planned. This study identifies a number of variables and variable-interactions that are believed to be important by a carefully selected group of domain experts. In relation to future research, a general conclusion is obviously that since the results come with quite a fair amount of uncertainty, certainty in the area of IDS effectiveness is lacking. The uncertainty expressed by the domain experts suggests that there are several nuisance variables that are not included in the (rather simple) model used in this study. This should be considered in future research.

The respondents of the survey were asked to suggest variables that were perceived as important for effectiveness by indicating which of the studied variables they would like to replace it with. Suggestions made were to add anomaly-based intrusion detection and to further detail the variable relating to the exploited vulnerability type. It is also likely that less uncertainty concerning the value of nuisance variables would reduce the uncertainty in domain experts' estimates. Thus, further studies involving domain experts could be employed to produce more precise hypotheses concerning effectiveness.

Another interesting finding is that the variables are rather independent (except the usage of HIDS and NIDS in combination). This seems to suggest that future research can reasonably be optimized in each variable domain independently.

For example in experiments concerning effectiveness or in repeated studies using a similar format to the one presented here.

As mentioned above – the survey also found relations that are contradicting well accepted causalities within the security community: that signature based detection can detect novel attacks and that a combination of HIDS and NIDS often is ineffective. These results are thus particularly interesting to investigate further. To some extent they also point in the same direction as the variable substitutions suggested by the respondents (anomaly based detection and vulnerability type).

7 Conclusion

Reliable data on intrusion detection effectiveness from observations or experiments expert is not available. The synthesized judgment of researchers in the intrusion detection field shows a great deal of uncertainty when estimating the effectiveness of IDSs for different scenarios. Some of this uncertainty stems from natural variation between enterprises. But it appears reasonable that a portion also come from epistemic uncertainty and strongly related to the lack of empirical studies in the field, i.e., the community is not certain on how well intrusion detection actually works.

This study provides indicators on the effectiveness of intrusion detection in different scenarios. In particular, host based solutions are associated with higher effectiveness than network based ones, tuning is a measure with comparably high impact on the effectiveness, and it is not of great importance to the effectiveness if the vulnerability exploited is well-known and patchable or if it is not. These quantitative results are based on the synthesized judgment of researchers in the field and indicate the importance of different variables and the effectiveness of solutions as a whole. If reliable data can be obtained from experiments or from observations of installed systems' effectiveness would allow tests of this result's validity.

9 References

[1] M. Sumner, “Information Security Threats: A Comparative Analysis of Impact, Probability, and Preparedness,” *Information Systems Management*, vol. 26, no. 1, pp. 2-12, Jan. 2009.

[2] S. Axelsson, “The base-rate fallacy and the difficulty of intrusion detection,” *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 186-205, Aug. 2000.

[3] J. P. Anderson, “Computer security threat monitoring and surveillance,” 1980.

[4] D. E. Denning, “An Intrusion-Detection Model,” *IEEE Transactions on Software Engineering*, vol. SE–13, no. 2, pp. 222-232, Feb. 1987.

[5] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems,” Gaithersburg, MD, USA, 2007.

[6] K. Salah and a. Kahtani, “Improving Snort performance under Linux,” *IET Communications*, vol. 3, no. 12, p. 1883, 2009.

[7] F. Alserhani, M. Akhlaq, I. U. Awan, J. Mellor, A. J. Cullen, and P. Mirchandani, “Evaluating Intrusion Detection Systems in High Speed Networks,” *2009 Fifth International Conference on Information Assurance and Security*, pp. 454-459, 2009.

[8] F. B. Ktata, N. E. Kadhi, and K. Ghédira, “Agent IDS based on Misuse Approach,” *Journal of Software*, vol. 4, no. 6, pp. 495-507, Aug. 2009.

[9] C. Xenakis, C. Panos, and I. Stavrakakis, “A comparative evaluation of intrusion detection architectures for mobile ad hoc networks,” *Computers & Security*, vol. 30, no. 1, pp. 63-80, Nov. 2010.

[10] R. Lippmann et al., “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation,” *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*, pp. 12-26, 1998.

[11] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, “The 1999 DARPA on-line intrusion detection evaluation,” *Computer Networks*, vol. 34, 2000.

[12] J. McHugh, “Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory,” *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262-294, Nov. 2000.

[13] B. I. A. Barry and H. A. Chan, “Intrusion detection systems,” in *Handbook of Information and Communication Security*, vol. 2001, no. 6, P. Stavroulakis and M. Stamp, Eds. Springer, 2010, pp. 193-205.

[14] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, “An overview of issues in testing intrusion detection systems, (NIST IR 7007),” Citeseer, 2003.

[15] T. Sommestad and A. Hunstad, “Intrusion detection and the role of the system administrator,” in *Proceedings of International*

Symposium on Human Aspects of Information Security & Assurance, 2012.

[16] R. Cooke and L. Goossens, "Expert judgement elicitation for risk assessments of critical infrastructures," *Journal of Risk Research*, vol. 7, no. 643–656, 2004.

[17] M. P. Krayer von Krauss, E. a Casman, and M. J. Small, "Elicitation of expert judgments of uncertainty in the risk assessment of herbicide-tolerant oilseed crops.," *Risk analysis : an official publication of the Society for Risk Analysis*, vol. 24, no. 6, pp. 1515-27, Dec. 2004.

[18] E. McFadzean, J.-N. Ezingeard, and D. Birchall, "Information Assurance and Corporate Strategy: A Delphi Study of Choices, Challenges, and Developments for the Future," *Information Systems Management*, vol. 28, no. 2, pp. 102-129, Apr. 2011.

[19] R. M. Cooke, "TU Delft expert judgment data base," *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May 2008.

[20] E. Biermann, "A comparison of Intrusion Detection systems," *Computers & Security*, vol. 20, no. 8, pp. 676-683, Dec. 2001.

[21] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Göteborg, Sweden, 2000.

[22] P. Garciateodoro, J. Diazverdejo, G. Maciafernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1–2, pp. 18-28, Feb. 2009.

[23] M. A. Faysel and S. S. Haque, "Towards Cyber Defense : Research in Intrusion Detection and Intrusion Prevention Systems," *Journal of Computer Science*, vol. 10, no. 7, pp. 316-325, 2010.

[24] A. Ashfaq, M. Robert, A. Mumtaz, M. Ali, A. Sajjad, and S. Khayam, "A comparative evaluation of anomaly detectors under portscan attacks," in *Recent Advances in Intrusion Detection*, 2008, pp. 351–371.

[25] S. Shaikh, H. Chivers, P. Nobles, J. Clark, and H. Chen, "Characterising intrusion detection sensors," *Network Security*, vol. 2008, no. 9, pp. 10-12, Sep. 2008.

[26] R. Werlinger, K. Hawkey, and K. Muldner, "The challenges of using an intrusion detection system: is it worth the effort?," *SOUPS '08 Proceedings of the 4th symposium on Usable privacy and security*, no. 1, 2008.

[27] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 366–375.

[28] R. S. Thompson, E. M. Rantanen, and W. Yurcik, "Network intrusion detection cognitive task analysis: Textual and visual tool usage and recommendations," in *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 2006, vol. 50, no. 5, pp. 669–673.

[29] R. S. Thompson, E. M. Rantanen, W. Yurcik, and B. P. Bailey, "Command line or pretty lines?: comparing textual and visual interfaces for intrusion detection," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007, p. 1205.

[30] T. Itoh, H. Takakura, A. Sawada, and K. Koyamada, "Visualization of Network Intrusion Detection Data," *IEEE Computer Graphics and Applications*, vol. 26, no. 2, pp. 40-47, 2006.

[31] J. R. Goodall, W. G. Lutters, and A. Komlodi, "Developing expertise for network intrusion detection," *Information Technology & People*, vol. 22, no. 2, pp. 92–108, 2009.

[32] a Fink, J. Kosecoff, M. Chassin, and R. H. Brook, "Consensus methods: characteristics and guidelines for use.," *American journal of public health*, vol. 74, no. 9, pp. 979-83, Sep. 1984.

[33] A. H. Ashton, "Does consensus imply accuracy in accounting studies of decision making?," *The Accounting Review*, vol. 60, no. 2, pp. 173–185, 1985.

[34] D. J. Weiss and J. Shanteau, "Empirical Assessment of Expertise," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 1, pp. 104-116, 2003.

[35] M. J. Abdolmohammadi and J. Shanteau, "Personal attributes of expert auditors," *Organizational Behavior and Human Decision Processes*, vol. 53, no. 2, pp. 158–172, 1992.

[36] J. Shanteau, D. J. Weiss, R. P. Thomas, and J. C. Pounds, "Performance-based assessment of expertise: How to decide if someone is an expert or not," *European Journal of Operational Research*, vol. 136, no. 2, pp. 253–263, 2002.

[37] R. Cooke, *Experts in Uncertainty: Opinions and Subjective Probability in Science*. New York, New York, USA: Open University Press, 1991.

[38] R. T. Clemen and R. L. Winkler, "Combining probability distributions from experts in risk analysis," *Risk Analysis*, vol. 19, no. 187, pp. 187-204, 1999.

[39] F. Bolger and G. Wright, "Assessing the quality of expert judgment: Issues and analysis," *Decision Support Systems*, vol. 11, no. 1, pp. 1-24, Jan. 1994.

[40] Elsevier B.V., "Scopus," 2011. [Online]. Available: http://www.scopus.com/.

[41] S. T. Cavusgil and L. A. Elvey-Kirk, "Mail survey response behavior: A conceptualization of motivating factors and an empirical study," *European Journal of Marketing*, vol. 32, no. 11/12, pp. 1165–1192, 1998.

[42] P. H. Garthwaite, J. B. Kadane, and A. O'Hagan, "Statistical methods for eliciting probability distributions," *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 680-701, 2005.

[43] L. J. Cronbach and R. J. Shavelson, "My Current Thoughts on Coefficient Alpha and Successor Procedures," *Educational and Psychological Measurement*, vol. 64, no. 3, pp. 391-418, Jun. 2004.

[44] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[45] R. Cooke, *Experts in uncertainty: opinion and subjective probability in science*. 1991.

[46] G. Lyon, "Nmap," 2011. [Online]. Available: http://nmap.org/.

[47] I. Sourcefire, "Snort::VRT," 2011. [Online]. Available: http://www.snort.org/vrt.

[48] I. Sourcefire, "Snort::Home page," 2011. [Online]. Available: http://www.snort.org/.

[49] U. S. D. of C. NIST Computer Security Resource Center, "National Vulnerability Database," 2011. [Online]. Available: http://nvd.nist.gov/.

[50] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Published by FIRST-Forum of Incident Response and Security Teams*, 2007, pp. 1-23.

[51] D. C. Montgomery, *Design and analysis of experiments*. John Wiley & Sons Inc, 2008.

[52] S. Lin, "A study of expert overconfidence," *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 711-721, May 2008.

Paper E:
Estimates of success rates of denial-of-service attacks

Teodor Sommestad, Hannes Holm and Mathias Ekstedt

Abstract

Denial-of-service (DoS) attacks are an imminent and real threat to many enterprises. Decision makers in these enterprises need be able to assess the risk associated with such attacks and to make decisions regarding measures to put in place to increase the security posture of their systems. Experiments, simulations and analytical research have produced data related to DoS attacks. However, these results have been produced for different environments and are difficult to interpret, compare, and aggregate for the purpose of decision making. This paper aims to summarize knowledge available in the field by synthesizing the judgment of 23 domain experts using an establishing method for expert judgment analysis. Different system architecture's vulnerability to DoS attacks are assessed together with the impact of a number of countermeasures against DoS attacks.

1 Introduction

Denial-of-service (DoS) attacks on information technology based services are a relatively common type of security incident and produce a substantial share of the losses incurred from attacks on information technology.

To manage the risk related to DoS attacks in practice, decision makers need to be able to understand and estimate the probability that their information technology based services can be disturbed by this type of attack. Hence, data on the probability of attack success given different conditions in the information technology infrastructure would contribute to more informed decision making when it comes to risks associated with DoS attacks.

There are literature that summarizes this problem domain and the potential of different countermeasures, for example, the review made by Peng et al. [1]. In this review, four categories of defense against DoS attacks are identified: attack prevention, attack detection, attack source identification, and attack reaction. All of these are relevant, however, this study only focus on the first type of defense – attack prevention.

There is plenty of research on techniques for attack prevention in terms of simulations, experiments, and analytical calculations. However, this research is difficult to use in a decision making situation. The simulations, experiments, and calculations are made for a specific configuration and aims to be representative for a specific context [2]. Therefore, unless the decision maker has this specific situation at hand, these results must first be interpreted and somehow synthesized before they can be used to answer questions related to the decision making situation at hand.

This paper aims to summarize knowledge that exists in the research community on how difficult it is to succeed with DoS attacks in general, and how effective different preventive countermeasures are against these attacks. This is done through a survey distributed to experts on DoS attacks. The experts were asked to estimate success probabilities in different scenarios. Since the scenarios were defined on a high level of abstraction, the answers from any expert would be inherently uncertain. In order to take this fact into account the answers were given as probability distributions of attack success. In order to arrive at as credible results as possible estimates of the experts were weighted using an established method for expert judgment analysis. Thus, in summary, the estimates are made for a number of selected system scenarios and show both expected effectiveness of countermeasures and the uncertainty of these estimates.

The rest of this paper is structured as follows. Section 2 presents related work and the scenarios for which success probabilities were assessed. Section 3 presents the method for expert judgment analysis, known as Cooke's classical method. Section 4 presents the data collection method. Section 5 shows the result. Section 6 discusses these results and their implications. Section 7 draws conclusions.

2 Studied Denial-of-Service attack scenarios

Denial-of-service (DoS) attacks can be divided in two types [3]. The first type, *semantic attacks*, causes DoS by sending carefully crafted packets to the targeted system (also known as software exploits [4]). These packets exploit vulnerabilities in the target system and make it unresponsive, e.g. by crashing the system. The second type, *brute force attacks*, occupies the target service with massive amounts of traffic that impairs it so that it cannot serve legitimate users (also known as flooding attacks [4]). This study covers both these classes of attacks. Previous work in both types of attack is presented below together with the variables included in this study. The selected variables have been chosen based on (1) relevance to practical applications and their usage in practice today, (2) their expected impact in the possibility to succeed with the attack and (3) their relevance to decision makers of software based services. Relevant variables have been selected based on a literature review. This selected variables relevance and the prioritizations made were validated by two external security professionals.

2.1 Semantic attacks

Software vulnerabilities are common in software products and many of these can be used to influence the availability of the vulnerable system. More than two thirds of known software vulnerabilities have an impact on availability [5], i.e., they can be used to cause DoS. There are several aspects that influence if an attacker can exploit the software vulnerability. The Common Vulnerability Scoring System [6] includes: the access vector that is possible to use (i.e. remotely exploitable or only locally exploitable), if the attacker must be able to bypass authentication before exploitation, and the ease of exploitation (e.g. if it is easy to construct the exploit code).

The most obvious countermeasure for this type of attack is to remove the software vulnerability, e.g., by updating the software to a version without the vulnerability. However, this is not always possible to do and is in the typical case associated with an effort and cost. Also, exploitation might be possible even if the software is without what would be regarded as software vulnerabilities per se. For example, by exploiting the intended functionality in an abusive way as when recursive payloads are sent to a web service [7].

There are also measures that influence the exposure that is experienced when software exhibits a software vulnerability. There are a number preventive measures for semantic attacks, for instance, in [8] a toolkit for defensive programming is presented. However, preventive measures these are seldom used in practice.

This study only investigates remote attacks. Hence, the investigated attack vector is remote exploitation. The model used to assess DoS attacks success rate includes three variables for semantic attacks (cf. Table 1). These are: (1) if the attacker can provide access credentials to the targeted system (*AC, Access Credentials)*, (2) the presence of a software vulnerability (SV, *Software Vulnerability*) in the target, and (3) the target of the DoS attack. For (3), the goal is to cause DoS for an entire machine or the target is to cause DoS on a specific service.

Table 1. Variables studied for semantic DoS attacks

Variable	Description
AC	Access credentials: if the attacker can authenticate itself as a legitimate user of the service.
SV	Software vulnerability: if the software has an implementation vulnerability.
Machine	If the DoS attack targets a machine (e.g. a CPU), or a specific service running on the machine.

2.2 Flooding attacks

A substantial amount of research has been spent on brute force attacks, in particular distributed DoS attacks. Excellent compilations of attack form within this category of attacks can be found in [3], [9], [1]. The taxonomy in [1] focus on preventive measures on the network level, e.g., ingress and egress filtering at internet service providers. While this certainly has an influence on the possibility to perform certain attacks, it is difficult to influence as a decision maker of software based services at their enterprise. In the taxonomy of [3] preventive measures against flooding attacks include: system security (e.g. to reduce botnets on the internet), protocol design, resource accounting, and resource multiplication. The first two of these are again difficult to influence as an enterprise decision maker and; the third can be seen as a reactive measure [1]. In addition to the abovementioned measures, the taxonomy given in [9] includes: changing IP address, honeypots, disabling unused services, and secure overlay services.

Based on the criteria given above the following variables were selected for this study: changing IP address through proactive server roaming [10], [11] and resource multiplication (i.e. redundancy) with load balancing [3].

Table 2. Variables studied for brute force DoS attacks.

Variable	Description
Roaming	The service uses proactive server roaming.
Load balancing	There is a load balancer in between the attacker and the target.

2.3 Assumptions

In addition to the variables given above a number of conditions were kept constant in the scenarios. The attacker is an outsider with the competence of a professional penetration tester who has access to tools that are free or commercially available. The attacker has spent one week preparing for the attack and the attack is performed from an external network. Also, in the case of brute force attacks it should be assumed that there is an enterprise firewall between the attackers host(s) and the targeted service. However, in all cases the attacker can reach the targets IP address and port.

Even with these assumptions the scenario definitions only covers a subset of the variables of relevance. They are also given on a coarse detail level. For instance, the details associated with the software vulnerability are not specified and the amount of redundancy implemented behind the load balancer. To avoid unnecessary ambiguity the respondents were asked to consider unspecified variables to be in the state they typically are in an enterprise environment. For instance, if enterprises often are protected by ingress and egress filtering this should be accounted for and considered in the estimates given. Any uncertainty caused by this should be reflected in the estimates.

3 Synthesizing expert judgments

There is much research on how to combine, or synthesize, the judgment of multiple experts to increase the calibration of the estimate used. Research has shown that group of individuals assess an uncertain quantity better than the average expert, but the best individuals in the group are often better calibrated than the group as a whole [12]. The combination scheme used in this research is the classical model of Cooke [13]. Experience shows that Cooke's classical method outperforms both the best expert and the "equal weight" combination estimates. In an evaluation involving 45 studies it performs significantly better than both in 27 studies and performs equally as well as the best expert in 15 of them [14].

In Cooke's classical method *calibration* and *information* scores are calculated for the experts based on their answers on a set of seed questions, i.e,. questions for which the true answer is known at the time of analysis. The calibration score shows how correct the respondent's answers match the true value; the information score shows how precise the respondent's answer are. These two scores are used to define a *decision maker* which assigns weights to the experts based on their performance. The weights defined by this decision maker are used to weight the respondents answer's to the questions of interest – in this case the operational scenarios described in section 2. In sections 3.1, 3.2 and in 3.3 Cooke's classical method is explained. For a more detailed explanation the reader is referred to [13].

3.1 Calibration score

In the elicitation phase the experts provide individual answers to the seed questions. The seed questions request the respondents to specify a probability distribution for an uncertain continuous variable. This distribution is typically specified by stating its 5th, 50th, and 95th percentile values. This yields four intervals over the percentiles *[0-5, 5-50, 50-95, 95-100]* with probabilities of p= *[0.05, 0.45, 0.45, 0.05]*. As the seeds are realizations of these variables the well calibrated expert will have approximately 5% of the realizations in the first interval, 45 % of the realizations in the second interval, 45 % of the realizations in the third interval and 5% of the realizations in the fourth interval. If s is the distribution of the seed over the intervals the relative information of s with respect to p is: $I(s, p) = \sum_{i=1}^{4} \ln(s_i/p_i)$.

This value indicates how surprised someone would be if one believed that the distribution was p and then learnt that it was *s*.

If N is the number of samples/seeds the statistic of $2NI(s, p)$ is asymptotically Chi-square distributed with three degrees of freedom. This is asymptotic behavior is used to calculate the calibration *Cal* of expert *e* as: $\mathrm{Cal}(e) = 1 - \chi^2_3(2N\, I(s, p))$

Calibration measures the statistical likelihood of a hypothesis. The hypothesis tested is that realizations of the seeds (*s*) are sampled independently from distributions agreeing with the expert's assessments (*p*).

3.2 Information score

The second score used to weight experts is the information score, i.e., how precise and informative the expert's distributions are. This score is calculated as the deviation of the expert's distribution to some meaningful background measure. In this study the background measure is a uniform distribution over the interval zero to one.

If b_i is the background density for seed $i \in \{1,\ldots,N\}$ and $d_{e,i}$ is the density of expert *e* on seed *i* the information score for expert *e* is calculated as:

$$\inf(e) = \frac{1}{N} \sum_{i=1}^{N} I(d_{e,i}, b_i)$$

In other words, the information score is the relative information of the expert's distribution with respect to the background measure. It should be noted that the information score does not reflect calibration and does not depend on the realization of the seed questions. So, regardless of what the correct answer is to a seed question a respondent will receive a low information score for an answer that is similar to the background measure, i.e., the answer is distributed evenly over the variable's range. Conversely, an answer that is more certain and focused the probability density over few values will yield high information scores.

3.3 Constructing a decision maker

The classical method rewards experts who produce answers with high calibration (high statistical likelihood) and high information value (low entropy). A strictly proper scoring rule is used to calculate the weights the decision maker should use. If the calibration score of the expert *e* is at least as high as a threshold value the expert's weight is obtained as: *w(e) = Cal(e)* Inf(e),* if the expert's calibration score is less than the threshold value α. If the experts calibration is less than α, the expert's weight is set to zero, a situation which is common in practical applications.

The threshold value α corresponds to the significance level for rejection of the hypothesis that the expert is well calibrated. The value of α is identified by resolving the value that would optimize a virtual decision maker. This virtual decision maker combines the experts' answers (probability distributions) based on the weights obtained at the chosen threshold value (α). The optimal level for α is where this virtual expert would receive the highest possible weight if it was added to the expert pool and had its calibration and information scored as the actual experts.

When α has been resolved the normalized value of the experts weights *w(e)* are used to combine their estimates of the uncertain quantities of interest.

4 Data Collection Method

This section presents how the survey data was collected by explaining: how seed questions for Cooke's classical method were assessed; which population and sample of experts that was chosen; how the measurement instrument was developed and tested.

4.1 Seed questions

As the experts performance on answering the seed questions are used to weight them, it is critical that the seeds are highly validated and also that they lie in the same domain as the studied variables. Thus, the seeds should represent the truth and it should be difficult to tell them apart from the questions of the study. They need to be drawn from the respondents' domain of expertise, but need not necessarily be directly related to questions of the study [13].

Naturally, the robustness of the weights attributed to individual experts depends on the number of seeds used. Experience shows that eleven seed questions are more than enough to see substantial difference in calibration [13]. This study used eleven seed questions to weight the respondents.

These eleven seed questions were of two types. The first type asked the respondents to estimate characteristics of known vulnerabilities related to DoS attacks. The correct answer was drawn from US Department of Commerce National Vulnerability Database [5]. The second type of question related to actual distributed DoS attacks of activity and how it influenced enterprises. The data for these questions came from the survey result presented in [15]. Summaries of the actual questions are presented in Table 3.

Table 3. Seed Questions.

#	Question	Value (%)
1	What is the share of known vulnerabilities with some impact on availability?	71
2	Of the known vulnerabilities with some impact on availability, how large portion can be exploited from external networks?	85
3	Of the known vulnerabilities with some impact on availability, how large portion requires that the attacker can bypass authentication?	5
4	What is the share of known vulnerabilities with some impact on availability that affect Windows 7?	85
5	What is the share of known vulnerabilities with complete impact on availability?	23
6	What portion of organizations in EMEA and US that operate their business online has an important online reputation use some on-premise/in-house DDoS protection technology?	65
7	What portion of organizations in EMEA and US that operate their business online or have an important online reputation over provision their bandwidth to protect against potential DDoS threats?	28
8	What portion of organizations in EMEA and US that operate their business online, have an important online reputation or operate financial services are primarily suffering from target DDoS attacks and aware of whom the attackers are?	30
9	What portion of organizations in EMEA and US that operate their business online or have an important online reputation or operate online financial services is primarily suffering from random DDoS?	52
10	What portion of organizations in EMEA and US that operate their business online or have an important online have experienced a DDoS attacks during a year that did disrupt services?	31
11	What portion of organizations in EMEA and US that operate their business online, has an important online have experienced and has experienced DDoS attacks needed more than 5 hours to recover from the most severe attack?	41

4.2 The domain experts

Studies of expert’s calibration have concluded that experts are well calibrated in situations where with learnability and with ecological validity [16]. Learnability comes with models over the domain, the possibility to express judgment in a coherent quantifiable manner that could be verified, and the opportunity to learn to from historic predictions and outcomes. Ecological validity is present if the expert is used to making judgments of the type they are asked for.

This study asks questions on the success of attempted DoS attacks, given different conditions. These judgments can be expressed in a quantifiable coherent and quantifiable manner. Persons with experience in DoS attacks (directly or indirectly) will also have access historic outcomes to learn from. Good candidates for this are researchers and penetration testers in the security field. These can be expected to both reason in terms of success or failure of an attacks in different condition. They also make such judgments in their line of work and evaluate different options (i.e., there is ecological validity). DoS attack researchers were therefore chosen as the population to survey.

To identify suitable security researchers articles published in the SCOPUS [17], INSPEC or Compendex [18] databases between January 2005 and September 2010 were reviewed. Authors who had written articles in the information technology field with any of the words “denial of service attack” or “denial-of-service attack” in the title, abstract, or keywords were identified. If their contact information could be found they were added to the list of potential respondents, resulting in a sample of 1378 respondents. After reviewing and screening respondents and their contact information a sample of 1065 individuals was assessed. Of these the used contact information to at approximately 180 turned out to be incorrect or outdated.

Out of approximately 885 researchers invited to the survey 296 opened the survey and 65 submitted answers to questions in the survey. A response rate of this magnitude is reasonable to expect from a slightly more advanced survey as this. Consistency checks and completeness checks were used to ensure the quality of answers used in the analysis. After these controls 23 respondents’ answers remained and these 23 were used in the final analysis.

As recommended by [19], motivators were presented to the respondents invited to the survey: i) helping the research community as whole, ii) the possibility to win a gift certificate on literature, and iii) being able to compare their answers to other experts after the survey was completed.

4.3 Elicitation instrument

A web survey was used to collect the probability distributions from the invited respondents. The survey was structured into four parts, each beginning with a short introduction to the section. First, the respondents were given an introduction to the survey as such that explained the purpose of the survey and its outline. In this introduction they also confirmed that they were the person who had been invited and provided information about themselves, e.g., years of experience in the field of research. Second, the respondents received training regarding the answering format used in the survey. After confirming that this format was understood the respondents proceeded to its third part. In the third part both the seed questions and the questions of the study were presented to the respondents. Finally, the respondents were asked to provide qualitative feedback on the survey and the variables covered by it.

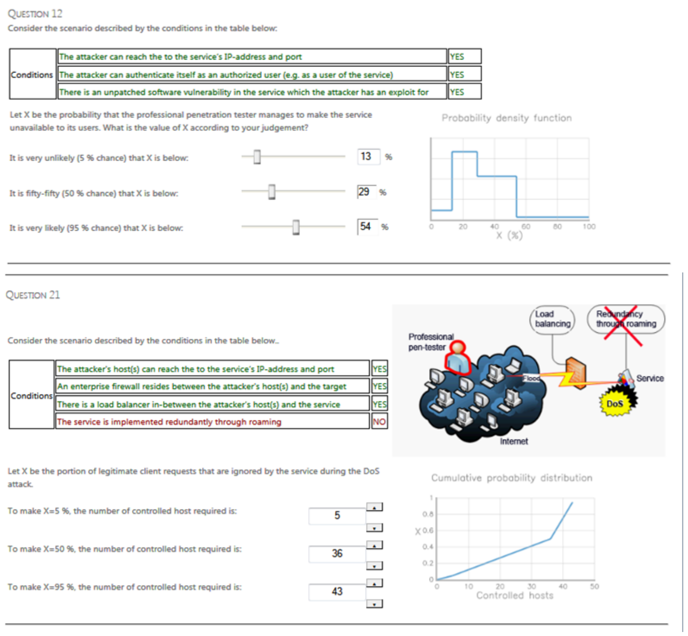
Questions in section three were each described through a scenario entailing a number of conditions. Scenarios and conditions for the seed questions can be found in Table 3; scenarios and conditions for the questions of interest in this study is described in section 5.

In the seed questions and the questions on semantic attacks the respondent was asked to provide a probability distribution that expressed the respondent's belief. As is custom in applications of Cooke's classical method this probability distribution was specified by setting the 5th percentile, the 50th percentile (the median), and the 95th percentile for the probability distribution. In the survey the respondents specified their distribution by adjusting sliders or entering values to draw a dynamically updated graph over their probability distribution. The three points specified by the respondents defines four intervals over the range [0, 100]. The graphs displayed the probability density as a histogram, instantly updated upon change of the input values.

In the questions concerning brute force attacks, the respondent also specified a probability distribution through the 5th, 50th and

95th percentile. However, they now specified the number of hosts the attacker would need to control to make 5, 50 or 95 percent of the legitimate requests being dropped. As before the estimates dynamically updated a graph representing the answer.

Use of graphical formats is known to improve the accuracy of elicitation [20]. Figures and colors were also used to complement the textual formulations and make the content easier to understand. In Figure 1 the format presented to respondents is exemplified.



QUESTION 12

Consider the scenario described by the conditions in the table below:

Conditions	The attacker can reach the to the service's IP-address and port	YES
	The attacker can authenticate itself as an authorized user (e.g. as a user of the service)	YES
	There is an unpatched software vulnerability in the service which the attacker has an exploit for	YES

Let X be the probability that the professional penetration tester manages to make the service unavailable to its users. What is the value of X according to your judgement?

It is very unlikely (5 % chance) that X is below: 13 %

It is fifty-fifty (50 % chance) that X is below: 29 %

It is very likely (95 % chance) that X is below: 54 %

QUESTION 21

Consider the scenario described by the conditions in the table below..

Conditions	The attacker's host(s) can reach the to the service's IP-address and port	YES
	An enterprise firewall resides between the attacker's host(s) and the target	YES
	There is a load balancer in-between the attacker's host(s) and the service	YES
	The service is implemented redundantly through roaming	NO

Let X be the portion of legitimate client requests that are ignored by the service during the DoS attack.

To make X=5 %, the number of controlled host required is: 5

To make X=50 %, the number of controlled host required is: 36

To make X=95 %, the number of controlled host required is: 43

Figure 1. Example of questions and answering formats used in the survey.

Elicitation of probability distributions is associated with a number of issues [20]. Effort was therefore spent on ensuring that the measurement instrument held sufficient quality. Before distribution of the survey the used question format as such had been tested in a pilot study on other security parameters. In that pilot study a randomized sample of 500 respondents was invited; 34 of these completed the pilot during the week it was open. The questions in this pilot survey were presented in the same way as in the present survey. A reliability test using Cronbach's alpha [21], [22] was carried out using four different ways to phrase questions for one variable. Results from this test showed a reliability value of 0.817, which indicates good internal consistency of the instrument.

5 Results

This section presents the result of the analysis performed on the judgment of the 23 experts. In section 5.1 the overall performance of the respondents on the seed questions is presented. In section 5.2 the synthesized estimates of those respondents who were assigned weight are presented.

5.1 Respondents' performance

As in many other studies involving expert judgment many of the experts were poorly calibrated on the seed questions. Their calibration score varied between $3.853*10^{-11}$ and 0.3697 with a mean of 0.0375; their information score varied between 0.222 and 1.974 with a mean of 1.00.

Cooke's classical method aims is to identify those respondents whose judgment is well calibrated and informative. The virtual decision maker was optimized at a significance level (α) of 0.1317. This meant that two experts were assigned a weight. They received weights 0.5288 and 0.4712 after normalization. As noted above it is not uncommon that a substantial number of respondents receive the weight zero with this method. The aim is to identify those respondents that are likely to be well calibrated on the questions at issue.

5.2 Success rate in the scenarios

The respondents' weights were used to construct the estimates on denial of service attacks' success rate given different conditions, i.e., the weighted mean of their distributions was calculated. The estimated distributions were assumed to be distributed in the same way as they were presented to the respondents, i.e., as depicted in Figure 1. Note that certain variables are kept constant over the scenarios, c.f. section 2.3.

5.2.1 Semantic attacks

As depicted in Table 4 the synthesized estimates show clear differences among the scenarios. The median for the scenarios varies between 16 and 76 percent; the value at the 5th percentile varies between 2 and 32 percent; the value at the 95th percentile varies between 56 and 95 percent.

In general it is more difficult to cause DoS for a single service than it is to cause DoS for an entire machine. As expected it is also more difficult to cause DoS in scenarios where there is access controls restricting access and where there is no software vulnerabilities.

The estimates in Table 4 are on the same format as results from a factorial experiment investigating all possible combinations. The influence strength of variables and their interactions can be calculated by comparing the scenarios with each other. For instance, the mean influence a software vulnerability (*SV*) has can be assessed as the mean of pairwise difference between scenarios #1 and #3, #2 and #4, #5 and #7, and #6 and #8.

Table 4. Attack scenarios for semantic attacks.

#	Target	SV	AC	5% value	50% value	95% value	Expected value
1	Machine	Yes	Yes	0.32	0.76	0.95	0.72
2	Machine	Yes	No	0.14	0.56	0.80	0.53
3	Machine	No	Yes	0.22	0.62	0.94	0.60
4	Machine	No	No	0.05	0.37	0.69	0.38
5	Service	Yes	Yes	0.10	0.48	0.93	0.50
6	Service	Yes	No	0.08	0.25	0.67	0.30
7	Service	No	Yes	0.11	0.42	0.86	0.46
8	Service	No	No	0.02	0.16	0.56	0.21

The variable weights are depicted in Table 5. The values show the influence this variable, or variable combination, have on the success probability. The target and presence of a software vulnerability are most important. If a machine is targeted (and not a specific service alone) the probability of success increase by 19 percent on average; the increase that comes from a software vulnerability is 21 percent. If the attacker has access credentials it increases the success rate with about 10 percent on average. The variables are more or less independent. This can be seen from the low values associated with variable combinations. These show the impact these particular combinations have on the success probability. For instance, the combination of software vulnerability and access credentials has the joint effect on the expected value of minus two percent units. The joint effect in addition to their individual influence of 21 and 10 percent units is thus comparably small.

Table 5. Semantic attacks – influence of variables on the success rate.

Variable or variable combination	5% value	50% value	95% value	Expected value
Machine	+0.11	+0.25	+0.09	+0.19
SV	+0.12	+0.24	+0.24	+0.21
AC	+0.06	+0.12	+0.08	+0.10
Machine & SV	+0.06	-0.01	-0.04	-0.01
Machine & AC	+0.04	+0.05	-0.02	+0.03
SV & AC	-0.02	-0.02	-0.04	-0.02
Machine & SV & AC	+0.02	-0.01	-0.02	0.00

5.2.2 Brute force attacks

Table 6 lists the estimates for brute force attacks in terms of the number of hosts required to attain a certain level of unavailability for users, i.e., 5, 50 and 95 percent ignored legitimate traffic. An intrinsic interval [13] of 10 percent was used to estimate the expected number of host required to denial a legitimate user access.

Table 6. Attack scenarios for brute force attacks – hosts required to cause unavailability.

#	Roaming	Load balancing	5% unav.	50% unav.	95% unav.	Expected value
1	Yes	Yes	15	30	58	33
2	Yes	No	15	21	47	25
3	No	Yes	15	26	43	26
4	No	No	10	18	38	20

The variable weights derived from these scenarios are shown in Table 7. Both load balancing and roaming has an effect on the number of host required. The joint effect is marginal also here. To have both at the same time only increase the expected number of hosts required with one, in addition to their individual effects.

Table 7. Brute force attacks – influence of variables on hosts required to cause unavailability.

Variable or variable combination	5% value	50% value	95% value	Expected value
Load balancing	+2.5	+3.5	+12	+6
Roaming	+2.5	+8.5	+8	+7
Load balancing & Roaming	-2.5	+0.5	+3	+1

6 Discussion

The method used to analyze the experts' judgments and combine these is discussed in section 6.1 below. The elicitation instrument used is discussed in section 6.2. The result as such and the importance variables included in the study are discussed in in section 6.3.

6.1 The expert judgment analysis

In this study Cooke's classical method [13] was used to synthesize expert judgments. This performance based method aims to select the experts that are well calibrated and combine their judgments in an optimal way. The track record of this method [14] positions it a best-practice when it comes to combining eliciting expert judgment of uncertain quantities.

Eleven seed questions were used to evaluate calibration and information scores. These seed questions are of two types. The first type of seed questions is drawn from a vulnerability database [5]. The second type is drawn from a survey on brute force attacks [15]. They have an obvious relation to the questions of interest and are therefore suitable for rating the respondents.

A concern to the validity is that these sources are available to the respondents who could have used them to identify the answers to the seed questions. If they would do so these seeds would not work well as a gauge for how well calibrated and informative the expert's own judgment is. However, it is unlikely that anyone did so. None of the respondents answering the survey has given comments that indicate that they have realized that the correct answer can be found in online databases or in publications. Also, the uncertainty expressed in their answers suggests that they did not base them directly on these sources.

The answers on the seed questions show that many experts in the field are poorly calibrated, i.e., their estimates do not match empirical observations well. Two respondents were assigned weight when the virtual decision maker was optimized. It is appropriate to perform robustness test of the solution when applying Cooke's classical method [13]. These are made with respect to both seed variables experts by removing one at a time and investigating the impact of this removal [13]. Such tests were performed and no undue influence was identified.

Experts are better at estimating quantities in domains where they are possible to learn from observations, e.g. from experiments or simulations [16]. In the survey the respondents were asked to state from where they had obtained the knowledge used to answer the survey's questions. Of the 22 respondents whose assessment was analyzed 10 had defended systems in practice, 20 had learnt from simulations, 22 had learnt from literature and 9 had learnt it from experiments. The two respondents receiving weight from Cooke's classical method had defended systems, learnt from simulations, and learnt from literature.

6.2 Validity and reliability of the elicitation instrument

Cooke [13] suggests that seven guidelines used when data is elicited from experts: (1) formulate clear questions, (2) use an attractive format for the questions and a graphical format for the answers, (3) preform a dry run, (4) have an analyst present during the elicitation, (4) prepare an explanation of the elicitation format and how answers will be processed, (6) avoid coaching and (7) keep elicitation sessions to less than one hour long.

This study follows with all these guidelines except (4) – to have an analyst present during elicitation. The invited researchers were given contact information to the research group when invited to the survey which they were encouraged to use any if questions arose. However, it is possible that analysts' physical absence suppressed some potential issues from being brought up during elicitation. The respondents were asked to comment the clarity of the questions and the question format used in the survey. Two respondents indicated that they had difficulties with answering in the format used while several others stated that the format was clear and understandable. The two respondents who had difficulties would have preferred a format without probability distributions instead; an ordinal rating was suggested instead. While this probably would make the questions easier to answer it would also be less expressive and more difficult to interpret.

6.3 Variables importance to the success rate

This study investigated three variables related to semantic attacks and two variables related to brute force attacks.

With respect to sematic DoS attacks the result indicates that it is easier to cause DoS for an entire machine than it is to cause DoS in a specific service. The increase on the success rate is on average 20 percentiles, which increase on the success rate with about 50 percent on average. The same magnitude of influence comes from to the existence of software vulnerabilities. If the attacker can authenticate itself to the target this increase the success probability with approximately 10 percent units. Removing software vulnerabilities and implementing access control that protects service's functionality against illegitimate users are two measures that can be implemented by decision makers. Together they would decrease the success probability with about 30 percent units and thereby reduce the probability of success to about half of what it would be without these measures.

With respect to brute force attacks, e.g., distributed DoS attacks, load balancing and roaming both increase the requirements placed on the attacker. Together they increase the number of hosts required to succeed with DoS by about 50 percent. Looking at the confidence intervals in Table 7 it also appears as if load balancing primarily help to protect against a complete DoS (c.f. the 95 percent value in Table 7), but it has less impact on the number of hosts required to make some users experience unavailability.

The scenarios estimated in this study do not detail all variables of relevance. As this was the case the respondents were asked to provide probability distributions representing the values for typical enterprises. If variations exists between enterprises (e.g. in terms of other protection mechanisms, hardware capacity, etc.) this should be accounted for in the estimates and thereby spread the estimated distributions over larger intervals. Judging from the span of the intervals on semantic attacks there are possibilities to increase (or decrease) the defense with other variables than the one included here. For instance, for the five percent of best defended systems the success probability of semantic attacks is below two percent, given that software vulnerabilities are removed, access controls are between the attacker and the target is a specific service (see #8 in Table 7). Conversely, the success probability for the same scenario is above 56 percent for the least defended five percent. How much of this uncertainty that arise from epistemic uncertainty and how much that arise from variations between enterprises is difficult to know. But it appears likely that both contribute to the uncertainty reflected in the estimates.

The variables included in this study were selected based on literature with the assistance of domain experts. To narrow the intervals and allow more precision, further variables need to be included in the scenarios' definitions. The respondents of the survey were asked to suggest other variables that they would like to replace the selected variables with. The suggestions were diverse, which suggests that the most significant factors were included. The full list of suggestions of variables included: defining if it is forced or strict load balancing, the amount of redundancy used by the load balancer, adjustments of the load balancer, routing schemes, the number of requests the target is designed for, and bandwidths of connections. Further work could explore these variables impact and produce narrower probability distributions. Based on the result presented here it appears as if the influences of the studied variables are independent. This could be valuable input to further work on this field.

7 Conclusion

This research generalizes quantities related to DoS attacks using expert judgment available in the research community and present approximate estimates on attackers ability cause DoS. The result shows the weight of key factors in semantic attacks and brute force attacks. Applying measures that are included in this research does have a significant impact on the success rate for semantic attacks and the number of controlled host required for a brute force attack. However, the result also shows the variation that is expected to be found between enterprises solutions through the probability intervals produced. The cause of these intervals is likely to arise because from a number of factors. The impact of other factors and their influence on the success of DoS attacks could be investigated in further work. This could include investigations of how large the epistemic uncertainty is about the actual values, i.e., how precise the research community's knowledge is on DoS attacks and factors that influence their success.

8 References

[1] T. Peng, C. Leckie, and K. Ramamohanarao, “Survey of network-based defense mechanisms countering the DoS and DDoS problems,” *ACM Computing Surveys*, vol. 39, no. 1, p. 3-es, 2007.

[2] R. Chertov, S. Fahmy, and N. B. Shroff, “Emulation versus simulation: A case study of TCP-targeted denial of service attacks,” in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, 2006, p. 10–pp.

[3] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, p. 39, Apr. 2004.

[4] A. Hussain, J. Heidemann, and C. Papadopoulos, “A framework for classifying denial of service attacks,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, p. 99110.

[5] NIST, “National Vulnerability Database Home Page,” 2010. [Online]. Available: http://nvd.nist.gov/. [Accessed: 16-Jun-2010].

[6] P. Mell, K. Scarfone, and S. Romanosky, “A complete guide to the common vulnerability scoring system version 2.0,” in *Published by FIRST-Forum of Incident Response and Security Teams*, 2007, pp. 1-23.

[7] A. Vorobiev and J. Han, “Security Attack Ontology for Web Services,” *2006 Second International Conference on Semantics, Knowledge, and Grid*, pp. 42-42, Nov. 2006.

[8] X. Qie, R. Pang, and L. Peterson, “Defensive programming: Using an annotation toolkit to build DoS-resistant software,” *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 45–60, 2002.

[9] C. Douligeris, “DDoS attacks and defense mechanisms: classification and state-of-the-art,” *Computer Networks*, vol. 44, no. 5, pp. 643-666, Apr. 2004.

[10] S. M. Khattab, C. Sangpachatanaruk, R. Melhem, and T. Znati, “Proactive server roaming for mitigating denial-of-service attacks,” in *Information Technology: Research and Education, 2003. Proceedings. ITRE2003. International Conference on*, 2003, pp. 286–290.

[11] C. Sangpachatanaruk, S. M. Khattab, T. Znati, R. Melhem, and D. Mossé, “A simulation study of the proactive server roaming for mitigating denial of service attacks,” in *Proceedings of the 36th annual symposium on Simulation*, 2003, p. 7.

[12] R. T. Clemen and R. L. Winkler, "Combining probability distributions from experts in risk analysis," *Risk Analysis*, vol. 19, no. 187, pp. 187-204, 1999.

[13] R. Cooke, *Experts in uncertainty: opinion and subjective probability in science*. 1991.

[14] R. Cooke, "TU Delft expert judgment data base," *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May. 2008.

[15] Forrester Consulting, "The trends and Changing Landscape of DDoS Threats and Protection," *Study on behalf of VeriSign, Inc*, 2009.

[16] F. Bolger and G. Wright, "Assessing the quality of expert judgment: Issues and analysis," *Decision Support Systems*, vol. 11, no. 1, pp. 1-24, Jan. 1994.

[17] Elsevier B.V., "Scopus," 2011. [Online]. Available: http://www.scopus.com/.

[18] Elsevier Inc, "Engineering Village," 2011. [Online]. Available: http://www.engineeringvillage.com. [Accessed: 24-Feb-2011].

[19] S. T. Cavusgil and L. A. Elvey-Kirk, "Mail survey response behavior: A conceptualization of motivating factors and an empirical study," *European Journal of Marketing*, vol. 32, no. 11/12, pp. 1165–1192, 1998.

[20] P. H. Garthwaite, J. B. Kadane, and A. O'Hagan, "Statistical methods for eliciting probability distributions," *Journal of the American Statistical Association*, vol. 100, no. 470, pp. 680-701, 2005.

[21] L. J. Cronbach and R. J. Shavelson, "My Current Thoughts on Coefficient Alpha and Successor Procedures," *Educational and Psychological Measurement*, vol. 64, no. 3, pp. 391-418, Jun. 2004.

[22] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

Paper F: The Cyber Security Modeling Language: A Tool for Vulnerability Assessments of Enterprise System Architectures

Teodor Sommestad, Mathias Ekstedt and Hannes Holm

Abstract

The Cyber Security Language (CySeMoL) is a modeling language for enterprise-level system architectures coupled to a probabilistic inference engine. If an enterprise's system architecture is modeled with CySeMoL this inference engine can assess the probability that attacks can be made against the components of the system architecture.. The theory embedded in CySeMoL, used for the attack probability calculations, is compilation of research results within a number of security-domains and covers a range of attacks and countermeasures. The theory has been validated on both a component-level and on a system-level. A test shows that the reasonableness and correctness of CySeMoL's assessments are comparable to those of a security professional. Its practical utility has been tested in case studies.

1 Introduction

Security issues related to information technology (IT) continue to be a concern in today's society. The IT environments of many enterprises are composed of a large number of systems connected to form a complex system-of-systems. Security is also a complex problem that is difficult to master. To fully estimate the security of an enterprise's system architecture, a large number of issues must be considered. Enterprise systems security managers must be able to assess how the vulnerabilities in one system influence the vulnerabilities in other systems. In addition, security managers must be able to assess how individual vulnerabilities influence the security of the entire system-of-systems, given the protection solutions that are used in different locations in the architecture.

Enterprise systems security managers typically have a basic understanding of their organization's architecture and systems and the losses incurred if assets are compromised. However, the managers' understanding of how vulnerabilities depend on each other in the system-of-systems and how the vulnerabilities can be exploited is often hazy. Support from security theory can be obtained from security experts and the literature. However, consulting security experts and studying the literature is both costly and time-consuming. Generally, support is missing for informed decision making concerning security on the system-of-systems level.

Tools that help system-security managers to assess how vulnerabilities in one system influence the vulnerabilities of other systems in enterprise system architecture are valuable, particularly if these tools can offer support without requiring input data that are difficult to collect.

1.1 Contribution

This paper presents an analysis tool called the Cyber Security Modeling Language (CySeMoL). This tool is built on the framework presented in [1] and uses a probabilistic relational model (PRM) [2] to support system-security managers in security analysis. If an object model of the system architecture is created according to a predefined class model, the tool can approximate the probability that an attacker will succeed with different attacks against the system. Security expertise is not required to create the object model because the PRM specifies a theory on how attributes in the object model depend on each other. The users must only model their system architecture and properties.

The theory used in CySeMoL is based on logical relations, experimental research in the security domain, and domain experts' judgment. CySeMoL covers a variety of attacks, including software exploits, flooding attacks, abuse of obtained privileges, and social-engineering attacks. Emphasis has been placed on supporting security managers concerned with attacks on industrial control systems (also known as Supervisory Control and Data Acquisition (SCADA) systems). However, the tool can be used for other types of domain.

This paper presents CySeMoL's PRM and the validation of this PRM. The PRM has been validated on the component and system levels. On the component level, the variables and relationships have been validated using the literature and domain experts. On the system level, the content validity has been tested by comparing the PRM's output with the responses of five security experts to a number of scenarios. In addition, the usability of the tool is demonstrated in two case studies at large enterprises.

1.2 Outline

Section 2 presents related works. Section 3 briefly describes the framework presented in [1], on which this tool is built. Section 4 presents the method used to create the tool. Section 5 presents CySeMoL's PRM. Section 6 presents the results of validity tests. Section 7 summarizes the paper and discusses future work.

2 Structured methods for security assessment

A substantial number of methods have been developed to quantify security and to support decision making related to security. For instance, Verendel [3] reviewed more than 100 methods for security metrication. The review presented below will cover only a subset of these methods. Emphasis is placed on methods that are applicable to assessing the security of a system-of-systems.

A number of prominent assessment methods require that the user is a security expert. For instance, the IEEE standard 27000-4 [4] and NIST's security metric guide [5] are methods that describe how an organization should develop and maintain a measurement program. However, the methods do not prescribe the measurements that should be taken or explain what different measurement values mean for security. These methods can be used as support when the security of a system-of-systems is assessed. However, they leave a substantial amount of effort to the user.

A number of methods offer concrete support and give the user a finished aggregation framework for security properties. Examples include the following: attack trees [6], defense trees [7], Boolean Logic Driven Markov Processes [8], the CORAS framework [9], Secure Tropos [10], and the model proposed by Breu et al. [11]. These methods help users combine variables to produce a meaningful result. Thus, the methods can help to combine the security values of single systems into a single value for a system-of-systems (i.e., the total risk). However, the methods require the user to produce the security ratings. For example, for defense trees, the user must quantify the likelihood of attacks being successful; for Boolean Logic Driven Markov Processes, provide time-to-compromise estimates; and in the model of Breu et al. give threat-realization probabilities. While some methodological support is available for quantification (e.g., [12]), expertise is still required. In addition, many of these methods require the users to identify causal dependencies in their systems, e.g., attack trees must be created before they can be used. For some systems, such causal models can be found in the literature, for example, the model employed [13] for electronic voting systems.

This paper describes a method that does not require security expertise from the user. In other words, the user of the method must only input information about the system architecture and is not required to provide security properties such as time-to-compromise or attack-success probability. Instead, the security properties for the system are derived from the system architecture and quantified according to a generic theory.

The practical utility of a method that quantifies the security of system architectures without requiring security expertise from the user is obvious. However, few methods of this type exist that are applicable to assessing the security of a system-of-systems. For instance, the Common Vulnerability Scoring System [14] produces assessments for a single software vulnerability, and the model presented in [15] produces assessments for single hosts.

Several of the methods that have a high abstraction level use best-practice standards to produce a security rating by the organization's compliance to the standard (e.g., Johansson's method [16]). The scope of such methods is useful where a system-of-systems is assessed. However, the ratings obtained are difficult to interpret and therefore not straightforward for system-security managers. For instance, knowing how high a value should be is difficult, as is deriving which risk is associated with a certain rating. Additionally, cause-and-effect relationships are not clear in these methods.

In recent years, a substantial number of articles have been published to develop methods that use attack graphs. An attack graph aims at determining which attacks can be conducted against a system. Because potential attacks are the source of cyber security risk, these methods fit well with decision-making processes concerning security. CySeMoL's approach is similar to the approach used in attack graphs.

Methods based on attack graphs are based on a model of the system architecture and a database of security exploits or security vulnerabilities [17], [18]. From these data, an algorithm calculates privileges and network states that can be reached by an attacker who starts from a given position [17].

Since the early variants of attack graphs appeared ([19],[20]), several tools have been developed that offer different solutions to the problem. Differences can be seen both in terms of the data they require as input and the output the produce when they are applied. The most mature tools are NETSPA [21], [22], MulVAL [23], and the TVA-tool [24]. These tools are described below.

NETSPA has been used to analyze networks of thousands of hosts, and its usability has been demonstrated in case studies [22]. However, NETSPA uses a coarse model of the attacker's capabilities. All software vulnerabilities in the database are considered to be exploitable by the attacker (given that the software can be reached) [21]. No differentiation is made with respect to the security measures implemented on the targeted host, to whether exploiting the vulnerability requires a particular configuration, or to the attacker's competence. GARNET [25] and its successor NAVIGATOR [26] build on NETSPA and add new visualization capabilities and support for what-if analysis.

MulVAL does not treat all vulnerabilities as unquestionably exploitable by the attacker. In MulVAL, each vulnerability is associated with a probability to represent how likely an attacker is to exploit the vulnerability [23]. This approach makes the model of potential attacks more expressive. Unfortunately, such probability values are not available in vulnerability databases. In descriptions of the method, the access-complexity rate from the US National Vulnerability Database has been translated into probability values [27]. However, no arguments are given for why this translation is used, and the validity of the translation remains unclear. Additionally, the probabilities only represent success rates generally and do not take into account protective measures that increase the difficulty of exploiting a vulnerability.

The TVA tool [24] uses a database of exploits possessed by the attacker instead of a database of vulnerabilities. The exploits are associated with detailed pre- and post-conditions, which state when the exploit can be applied and what state is reached after the exploit has been applied. Thus, the analysis can be constructed to represent an attacker armed with certain exploits. However, no database of exploits exists that is described this way. The data must be entered before use.

These three attack-graph methods offer different solutions to the problem of assessing the security of a given system specification. An issue all methods must address is the complex graphs that are produced when systems of realistic sizes are analyzed.

Additionally, they need to manage cycles in the graphs. Another issue is obtaining the input data. All three tools described above use the vulnerability scanner Nessus to collect these data. However, a recent accuracy test shows that Nessus misses more than half of the vulnerabilities when given access credentials to the hosts in a network and four out of five vulnerabilities when credentials are not given [28]. Thus, the automated scans on which the three tools rely are not reliable when individual vulnerabilities must be detected. In addition, in environments with sensitive systems (e.g., SCADA systems), scanners must be avoided because scanners can interrupt critical system services [29].

Another drawback of existing tools is the type of attacks that they cover. The tools are developed for software exploits targeting services running on the listening ports of machines. Thus, they lack the capability to model many relevant types of attack, e.g., password cracking, social engineering, and denial-of-service attacks. NETSPA has been extended to include attacks on clients (e.g., web browsers) [30]. However, the other two tools have not. Another matter falling outside the scope of these tools is zero-day exploits, i.e., attacks using vulnerabilities that are unknown to the public. The user of the tool can of course enter hypothetical data into the database and perform the analysis with these data. However, competence is required to identify which zero-day attacks can be reasonably expected from the attacker.

CySeMoL also models attacks and assesses the attacks that an attacker can execute. Compared with the three tools discussed above, CySeMoL analyzes a wider range of attack types and security measures. CySeMoL's output is probabilistic (as in MulVAL) and estimates the probability that different attacks can be accomplished against assets in the system architecture. The probabilities used in these calculations have been derived from experimental studies and studies eliciting the judgment of domain experts.

3 The Framework used: the PRM Template

The Cyber Security Modeling Language (CySeMoL) is built on the framework presented in [1]. This framework is a template for a probabilistic relational model (PRM) for security-risk analysis. Section 3.1 briefly describes the PRM formalism. Section 3.2 describes the security template for the PRMs presented. Section 3.3 describes the part of this template that is used in CySeMoL.

3.1 Probabilistic relational models

A PRM [2] specifies how a Bayesian network [31] should be constructed from an object model, i.e., how a Bayesian network should be created from a model that instantiates a class diagram, such as the one of the Unified Modeling Language (UML).

In a PRM, classes can have attributes and reference slots. The attributes are random variables with states from a discrete domain. The reference slots refer to other classes and express which relationships a class has with other classes. For instance, the attributes *System.Available* and *Person.Certified* could have the domain of values {True,False}. The reference slot *System.Administrator* could refer to the class *Person*.

The attributes in the PRM are associated with a set of parents. The parents of an attribute *A* are attributes in the object model that *A*'s value depends on. Parents are defined using a chain of reference slots that leads from the child attribute's object to the parent attribute's object. For instance, the attribute *System.Available* could be assigned the parent attribute *System.Administrator.Certified* using the reference slot *Administrator* of the class *System*. In this case, the slot chain is the single slot *System.Administrator*. Slot chains comprising multiple reference slots are also possible. If a slot chain points to attributes of more than one object in an instantiated model, an aggregate is used, e.g., one of the Boolean operators *OR* or *AND*.

Each attribute is associated with a conditional probability table that defines the attribute's value given all possible combinations of states in the attribute's parents. For instance, the attribute

System.Available would be given different probabilities that express the attribute's value when *System.Administrator.Certified* is *True* and *False*. The probabilistic model enables the value of attributes in an instantiated object model to be inferred. Such inference can also infer values for attributes with no assigned state.

In essence, a PRM defines how a Bayesian network shall be generated using the attributes of an object model. Thus, a PRM constitutes a formal machinery for calculating the probabilities of object properties in various architecture instantiations. For example, a PRM could be used to assess the availability of systems given that certain administrators are assigned to the systems.

3.2 The PRM template for security-risk analysis

In [1], a template for PRMs is defined for security-risk analysis. This template defines abstract classes together with attributes, reference slots and conceptual-attribute parents. The classes in this template are: *Asset*, *Owner*, *Threat*, *ThreatAgent*, *AttackStep*, and five types of *Countermeasure*. The countermeasures are: *ContingencyCountermeasure*, *PreventiveCountermeasure*, *DetectiveCountermeasure*, *ReactiveCountermeasure* and *AccountabilityCountermeasure*.

If a PRM is constructed according to this template and the PRM's conditional probabilities are assigned, the PRM can be used to perform two types of analysis. The first and more extensive analysis requires the instantiation of all the classes and can produce values for the expected economic losses for the architecture. This analysis includes consideration of the probability that different attack scenarios will be attempted and the expected loss that would be incurred if an attack is successful. The second analysis uses a subset of the template and can be used to calculate reachability values for different attack paths (threat instances), as in attack graphs. CySeMoL employs the second type of analysis.

3.3 The scope of the PRM

CySeMoL focuses on assessing the probability that attack paths can be accomplished given that they are attempted. Thus, CySeMoL uses a subset of the classes, attributes and dependencies defined in [1]. The class *AttackStep* is used to represent attacks together with the probability that the attacks are successful and that they are detected when they are attempted. The classes *PreventiveCountermeasure*, *DetectiveCountermeasure* and *ReactiveCountermeasure* are also included. Only one type of *ThreatAgent* is considered: a threat agent who has one week and publicly available tools.

CySeMoL's PRM includes attacks and countermeasures of relevance to industrial control and SCADA systems security. Threats against these systems are primarily related to the systems' availability and integrity properties (and not confidentiality). However, SCADA systems comprise the same type of subsystems as other information systems. Thus, the PRM can be used to analyze such systems but with limited support for threats against confidentiality. In addition, the PRM has other limitations, e.g., the countermeasures that the PRM can cover. These limitations are discussed in section 4.

4 The Method used to construct CySeMoL

This section presents the method used to construct CySeMoL, including a description of the qualitative structure (section 4.1) and the quantitative parameters associated with this structure (section 4.2). A summary is given in section 4.3.

Because this tool has been the subject of a considerable number of studies in the security field, this section does not describe each study in detail. The interested reader can find more information about these studies in the references.

4.1 Qualitative structure

The PRM's qualitative structure includes everything but the quantitative parameters, i.e., the classes, reference slots, attributes, and parents of attributes. CySeMoL's qualitative structure has been developed using the literature and judgment of security experts.

4.1.1 Literature study

The literature was studied extensively to identify an initial set of assets and which attack steps to include. This research included a review of a large number of textbooks (e.g., [32]), standards and reports (e.g., [29]), overviews (e.g., [33]) and security databases (e.g., [34]). Descriptions of attacks and countermeasures in these sources were used to create an initial model of a suitable level of abstraction and scope.

When the initial model was finished, the literature on specific attacks was consulted. This literature was used to determine the parents of attack steps, i.e., the countermeasures and privileges (completed attack steps) that influenced the probability that an attack step could be accomplished. A large number of sources were used for each type of attack. For instance, sources such as [35–39] were used to identify parents of remote code-exploitation attacks, and sources such as [40–42] were used to identify the parents of password-cracking attacks.

4.1.2 Review by domain experts

Before more detailed studies were conducted on specific attack types, the initial model was reviewed by three domain experts. All three were professional penetration testers. In interviews, these domain experts were asked to evaluate whether the model included the variables that are most useful when the security of a system-of-systems is to be assessed. To be useful, a variable should not only be important to security but also possible to assess. Thus, the experts were asked to consider which information was worthwhile to collect from a decision-making perspective. Several minor changes were made based on the suggestions by the experts. For example, firewall was modeled with fewer attributes and password protection was emphasized.

In addition to the general validation of the model, a number of domain experts were consulted on specific attacks and the parents to include for these attacks. Because such input requires a good understanding of the specific type of attack, different domain experts were used for different areas (cf. Table 1). For example, three persons were interviewed about intrusion-detection systems and the performance variables of such systems, and three persons were interviewed regarding remote arbitrary-code exploits. Few changes were suggested to the initial model during these validation efforts. Input from the domain experts helped to define variables in a practical way and to determine which variables to include (e.g., variables influencing the effort required to find new software vulnerabilities). Overall, the domain experts agreed with each other about variables relevance. This agreement suggests that the final model offered a good tradeoff between scope and usability.

4.2 Quantitative parameters

A PRM requires quantitative parameters for conditional probability distributions. CySeMoL's PRM consists of both logical dependencies with deterministic influences and probabilistic dependencies with uncertain influence. These dependencies are used to estimate the probability that a professional penetration tester can succeed with attacks against the architecture within one week using publicly available tools.

4.1.3 Logical, deterministic dependencies

A substantial portion (82 %) of the entries in the PRM's conditional probability tables is deterministic. These deterministic dependencies are used in the following cases:

a) Attack steps that are specializations of the same goal are aggregated into one attack step to simplify the model.
b) Certain preconditions are required for an attack to be possible.

The deterministic dependencies created for the first case are modeling constructs added for practical reasons. For example, denial-of-service attacks against services are decomposed into two variables representing two ways denial-of-service attacks can be conducted (semantic attacks or flooding attacks). This decomposition makes the conditional probabilities for each attack type easier to follow.

Deterministic dependencies of the second type are present when a condition is necessary to accomplish an attack step. For example, to perform remote code execution against a software service, two necessary (but not sufficient) conditions are that the attacker must be able to send data to the port the service listens to and that the service has a software vulnerability. The second type of deterministic dependency was identified from the literature and validated by domain experts in interviews. Examples of such dependencies are given in section 5.2.

4.1.4 Probabilistic, uncertain dependencies

Dependencies not determined by logical dependencies are uncertain and are defined using probabilities. Such dependencies are uncertain because the PRM does not include all the details that determine the variable's actual, which is the case if the PRM lacks a variable that could be important (e.g., the countermeasure application whitelisting) or if a variable's states represent a range of values (e.g., the severity rating of software vulnerabilities, divided into three levels). Such simplifications arise from the practical reasons discussed above, i.e., the creation of an instance model should not be costly.

Some probabilistic relationships could be specified based on published data from experiments and observations. For instance, data on the success of password cracking given different conditions could be found in [40–43]. However, for most of the conditional probabilities required, reliable quantitative data cannot be found in the literature. For instance, experiments on intrusion-detection systems are difficult to generalize from [44], and data on efforts required to find new software vulnerabilities are not gathered in a systematic way [45].

When reliable data could not be found in the literature, estimates were collected from domain experts. The data collected this way come from five surveys. The number of respondents to these surveys varies between four and 165 individuals. In four of the five surveys ([46–49]), the respondents' judgment was weighted using Cooke's classical method [50], a well-established method for weighting domain experts based on their ability to accurately assess a set of test questions on the same topic as the real questions. The effectiveness of the method is demonstrated in [50]. In the fifth study [51], the experts were weighted based on the number of real systems on which they had tested the variable's state.

4.2 Summary

The attacks and countermeasures included in CySeMoL were identified using the literature and input from domain experts. The aim of the qualitative structure was to be as complete as possible while remaining useful to a typical system-security manager. The quantitative parameters in the PRM are deterministic dependencies between attributes and uncertain dependencies between attributes. The probabilities for the uncertain dependencies are derived from observations of systems, experiment, and studies based on structured expert elicitation. An overview is given in Table 1.

Table 1. Overview of methods used

Part of the PRM	**Qualitative validation method**	**Parameterization method**
Discovering new vulnerabilities	Literature and 3 experts.	Cooke's classical method applied to 17 domain experts' judgment [46].
Remote arbitrary code exploitation attacks	Literature, pilot study, and 3 experts.	Cooke's classical method applied to 21 domain experts' judgment [47].
Intrusion detection	Literature and 3 experts.	Cooke's classical method applied to 165 domain experts' judgment [49].
Denial-of-service attacks	Literature and 2 experts.	Cooke's classical method applied to 23 domain experts' judgment [48].
Exploitation of network configuration mistakes	Literature and 2 domain experts.	Data described in [52] and [53] combined with four domain experts' judgment [51].
Attacks on password protection	Literature and one domain expert.	A review and synthesis of password-guessing data [40–42] and the capabilities of rainbow tables [43].
Social-engineering attacks	Literature.	Experiments [54–57] on social-engineering attacks.

5 The Cyber Security Modeling Language's PRM

This section describes the main contribution of this paper: CySeMoL. This section gives an overview of the metamodel (section 5.1), the deterministic and probabilistic dependencies embedded in the PRM (section 5.2), and the instantiation of attack paths (section 5.3). A full description of all concepts and dependencies is not given here because of the space the description would require. The description presented here gives an overview of CySeMoL and provides concrete examples of parts of the model. The interested reader can download the PRM and the software tool in which the PRM is implemented from [58].

5.1 Metamodel overview

The metamodel comprises 22 classes, 102 attributes, and 32 class relationships (reference slots). These classes, attributes, and class relationships dictate what information an architecture model should contain and are depicted in Figure 1. Two types of attribute are distinguished in Figure 1: countermeasures and attack steps. The upper box in a class describes the countermeasures associated with the class. The lower box describes the attack steps associated with the class. Relationships are marked by the dashed lines between classes.

The metamodel contains three concretized types of software: *OperatingSystem*, *Service*, and *ApplicationClient*. All types of *SoftwareInstance* are related to the *SoftwareProduct* the types are an installation of. An *OperatingSystem* can be related to a *NetworkZone* in which traffic between software instances is permitted (i.e., not filtered). The class *NetworkInterface* can connect multiple instances of *NetworkZone* and mark the instances as trusted or untrusted zones. A *NetworkInterface* can allow certain instances of *DataFlow*. The other data flows are assumed to be blocked. A *DataFlow* has a *Protocol,* and a *DataFlow* can read or write to a *DataStore* owned by a *SoftwareInstance*.

The *NetworkInterface* is related to the *Firewall* that enforces the *NetworkInterface*'s rules. A *Firewall* can be related to the class *DeepPacketInspection* and to an *IDSsensor* that enhances the *Firewall*'s capabilities. An *IDSsensor* can be associated with an *OperatingSystem* when the *OperatingSystem* is a host-based intrusion-detection system. A *DeepPacketInspection* can be associated with the *Service* on which the *DeepPacketInspection* focuses or the *ApplicationClient* for which it acts as a proxy.

All types of *SoftwareInstance* can be associated with an *AccessControlPoint,* which controls access to the software, e.g., a network *Service* or *OperatingSystem*. An *AccessControlPoint* is associated with an *AuthenticationMechanism,* which authenticates access requests and the *Account* instances that are allowed access. Because passwords are common, the two preceding classes have been specialized to *PasswordAuthenticationMechanism* and *PasswordAccount*. An *Account* is owned by a *Person,* and a *Person* can be covered by an *AwarenessProgram*.

A *ZoneManagementProcess* is related to a *NetworkZone* and describes how systems within the *NetworkZone* are managed, e.g., if the machines have been hardened.

5.2 Attribute dependencies

The 22 classes have 102 attributes. The attributes' values in an instantiated model are used to assess the security of the modeled enterprise systems. More precisely, the probability that certain attack paths (i.e., chains of attack steps) can be accomplished is used to assess the security of the architecture. This section gives examples of attack paths that may be instantiated in an instance model and the countermeasures that influence their probability of success.

If an attacker attempts to log on to a *SoftwareInstance,* the attacker may be required to bypass an *AccessControlPoint*, i.e., if the *SoftwareInstance* has a relationship to the *AccessControlPoint*. An *AccessControlPoint* is associated with an *AuthenticationMechanism* and *Account*s that belong to *Person*s who are authorized to access the system. An *Account*'s password may be compromised by being guessed online, offline, or through social engineering.

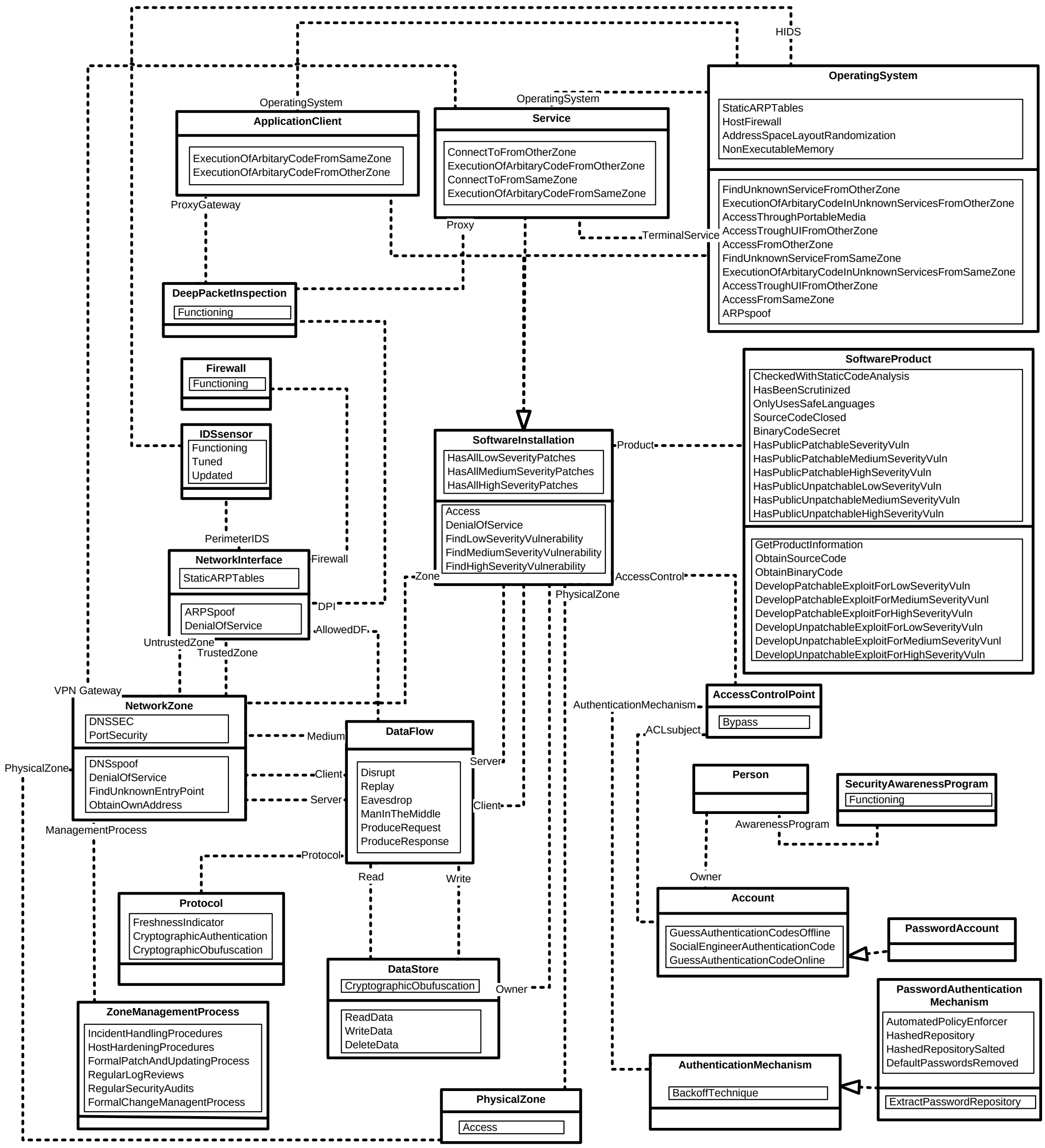

HIDS
OperatingSystem
StaticARPTables
HostFirewall
AddressSpaceLayoutRandomization
NonExecutableMemory
FindUnknownServiceFromOtherZone
ExecutionOfArbitaryCodeInUnknownServicesFromOtherZone
AccessThroughPortableMedia
AccessTroughUIFromOtherZone
AccessFromOtherZone
FindUnknownServiceFromSameZone
ExecutionOfArbitaryCodeInUnknownServicesFromSameZone
AccessTroughUIFromOtherZone
AccessFromSameZone
ARPspoof
OperatingSystem
ApplicationClient
ExecutionOfArbitaryCodeFromSameZone
ExecutionOfArbitaryCodeFromOtherZone
ProxyGateway
OperatingSystem
Service
ConnectToFromOtherZone
ExecutionOfArbitaryCodeFromOtherZone
ConnectToFromSameZone
ExecutionOfArbitaryCodeFromSameZone
Proxy
TerminalService
DeepPacketInspection
Functioning
Firewall
Functioning
IDSsensor
Functioning
Tuned
Updated
PerimeterIDS
NetworkInterface
StaticARPTables
ARPSpoof
DenialOfService
Firewall
DPI
AllowedDF
UntrustedZone
TrustedZone
SoftwareInstallation
HasAllLowSeverityPatches
HasAllMediumSeverityPatches
HasAllHighSeverityPatches
Access
DenialOfService
FindLowSeverityVulnerability
FindMediumSeverityVulnerability
FindHighSeverityVulnerability
Product
Zone
AccessControl
PhysicalZone
SoftwareProduct
CheckedWithStaticCodeAnalysis
HasBeenScrutinized
OnlyUsesSafeLanguages
SourceCodeClosed
BinaryCodeSecret
HasPublicPatchableSeverityVuln
HasPublicPatchableMediumSeverityVuln
HasPublicPatchableHighSeverityVuln
HasPublicUnpatchableLowSeverityVuln
HasPublicUnpatchableMediumSeverityVuln
HasPublicUnpatchableHighSeverityVuln
GetProductInformation
ObtainSourceCode
ObtainBinaryCode
DevelopPatchableExploitForLowSeverityVuln
DevelopPatchableExploitForMediumSeverityVunl
DevelopPatchableExploitForHighSeverityVuln
DevelopUnpatchableExploitForLowSeverityVuln
DevelopUnpatchableExploitForMediumSeverityVunl
DevelopUnpatchableExploitForHighSeverityVuln
VPN Gateway
NetworkZone
DNSSEC
PortSecurity
DNSspoof
DenialOfService
FindUnknownEntryPoint
ObtainOwnAddress
PhysicalZone
ManagementProcess
Medium
Client
Server
DataFlow
Disrupt
Replay
Eavesdrop
ManInTheMiddle
ProduceRequest
ProduceResponse
Server
Client
Protocol
Read
Write
AuthenticationMechanism
AccessControlPoint
Bypass
ACLsubject
Person
SecurityAwarenessProgram
Functioning
AwarenessProgram
Owner
Account
GuessAuthenticationCodesOffline
SocialEngineerAuthenticationCode
GuessAuthenticationCodeOnline
PasswordAccount
Protocol
FreshnessIndicator
CryptographicAuthentication
CryptographicObufuscation
DataStore
CryptographicObufuscation
ReadData
WriteData
DeleteData
Owner
PasswordAuthentication Mechanism
AutomatedPolicyEnforcer
HashedRepository
HashedRepositorySalted
DefaultPasswordsRemoved
ExtractPasswordRepository
ZoneManagementProcess
IncidentHandlingProcedures
HostHardeningProcedures
FormalPatchAndUpdatingProcess
RegularLogReviews
RegularSecurityAudits
FormalChangeManagentProcess
AuthenticationMechanism
BackoffTechnique
PhysicalZone
Access

Figure 1. Metamodel of CySeMoL's PRM. The upper box in a class contains the countermeasures associated with the class. The lower box contains the attack steps associated with the class.

Because password authentication is widely used, CySeMoL focuses on password authentication. The difficulty of compromising a *PasswordAccount* online depends on the presence of default passwords, whether password policies are automatically enforced, and whether the number of guesses that can be produced is limited.

The success of offline guessing depends on the possibility of extracting the password repository, whether password policies are automatically enforced, whether passwords are hashed, and whether passwords are salted. The success probability for social engineering is decreased if the account-owner is included in an *AwarenessProgram*.

Another way to obtain access to the *OperatingSystem* is to execute arbitrary code via *Services*, *ApplicationClients*, or services in the *OperatingSystem* that are unknown to the system owner. To accomplish these attack steps, the attacker must be able to connect to the *SoftwareInstance* in question, e.g., the attacked *Service* in question. Once connected, the attacker must accomplish a remote arbitrary-code attack.

To connect from another *NetworkZone,* the attacker must have access to a machine in the other *NetworkZone* and be able to send data over the *NetworkZone* that connects the two. Data can be sent if the attacker can produce a request in a *DataFlow* that has the *Service* as server or a response in a *DataFlow* that has the *ApplicationClient* as client. Requests and responses can be produced if the attacker has gained access to the software that produces the responses (e.g., an operating system) or by compromising the *DataFlow*'s integrity in a different way (e.g., by executing a man-in-the-middle attack using ARP-spoofing). To exploit unknown services in an *OperatingSystem,* the attacker must find such services. The probability that the attacker can find such services is influenced by attributes in the *ZoneManagementProcess* associated to the *OperatingSystem*'s *NetworkZone*.

To connect from the same zone, the attacker must obtain an own address in the *NetworkZone*. An address can be obtained by gaining access to an *OperatingSystem* in the zone, by breaking the *PhysicalZone* of the *NetworkZone* and connecting an own machine, or by finding an unknown entry point (e.g., an undocumented dual-homed computer) to the *NetworkZone*. The attributes in the *ZoneManagementProcess* influence the possibility the attacker may find unknown entries to a *NetworkZone*.

When an attacker can send data to the software, the attacker can attempt to execute arbitrary code remotely. The possibility of succeeding with this approach is influenced by the presence of address space layout randomization, non-executable memory protection, and whether the attacker has access rights to the software in question. If the attack is executed from another zone, the existence of deep-packet inspection in firewalls will have an influence. An *IDSsensor* will influence the possibility of detecting the attack. The influence of the *IDSsensor* depends on whether the attack is from the same *NetworkZone* or not.

The possibility of performing remote arbitrary-code exploits is also contingent on the existence of a high-severity software vulnerability in the target. The attacker may search the *SoftwareInstance*'s *SoftwareProduct* for publicly known vulnerabilities that have not been patched, or the attacker may invest time in searching for previously unknown vulnerabilities in the *SoftwareProduct*. The success rate of the former approach depends on the existence of known vulnerabilities and patches as well as the patching procedures in the *ZoneManagementProcess*. The success of the latter approach depends on attributes related to the *SoftwareProduct* (e.g., if developers have tested the *SoftwareProduct's* security using static analysis tools).

A denial-of-service attack against a *SoftwareInstance* can be accomplished if the attacker has access to the *SoftwareInstance's OperatingSystem*. Network-based denial-of-service attacks can be performed against a *Service* or *OperatingSystem* if the attacker can connect to the *Service* or *OperatingSystem*. Such attacks can also be conducted against a *DataFlow* if the attacker can accomplish unavailability in associated clients, servers, or mediating *NetworkZone*s.

5.3 Attack-path generation and assessment

An attack path is an ordered set of attack steps. Because the causal dependencies are expressed in CySeMoL's PRM, producing an exhaustive list of all attack paths that should be assessed is straightforward as long as the maximum number of steps is specified.

For each identified attack path, the corresponding Bayesian network is created and the success probabilities for all included attack steps are calculated. The attack steps in an identified attack path will be influenced by the attack steps in the path, the attack steps not in the path, and countermeasures in the system architecture. Attack steps not in the path are assigned a success probability of zero (because these steps are not attempted). All other attribute values are calculated as the PRM prescribes.

6 Verification and Validation

CySeMoL can be viewed as an expert system that assesses attack paths in a system architecture and estimates the probability that different attack paths can be traversed by a professional penetration tester within one week. The correctness and accuracy of this estimate is essential for the practical utility of CySeMoL. This section describes the verification and validation of CySeMoL based on the terminology and recommendations of [59].

6.1 Verification

Verification concerns the consistency, completeness, and correctness of the software implementation of the expert system [59]. A verification procedure can either be domain dependent and check for anomalies in the system using meta-knowledge on what is typical in the domain, or the verification procedure can be domain independent and look for general anomalies and errors in the implementation [59].

Domain-independent verification has been performed through inspections of the output produced by the tool. The result produced for both fictive test cases and real architectures has been inspected. These checks ensured that all attack paths that should be created by the model are present in the output and that the model does not contain redundant attack paths. The checks also verified that changes to a system's architecture produce the results prescribed by the theory in the model.

Domain-independent verification has focused on inspecting whether the PRM implementation is consistent (e.g., in naming attributes), complete (i.e., that all attribute parents are included), has the correct weights (i.e., the conditional probabilities), and infers data correctly (i.e., attack-generation procedure). CySeMoL is implemented as an extension to the Enterprise Architecture Analysis Tool (EAAT) [60]. EAAT implements PRM-inference with the SMILE library used in Genie [61]. Thus, the probabilistic inference mechanism has been verified in other projects.

6.2 Validation

An expert system's validity should be assessed in relation to a criterion [59]. CySeMoL has been validated using the criterion that CySeMoL should have expertise similar to that of a security expert.

Validity tests can be performed on a component level to validate pieces of the expert system or on a system level to validate the full expert system against the criterion. CySeMoL has been validated on both levels.

On a component level, CySeMoL has been validated by domain experts in interviews and surveys. As described in section 4, these experts have validated the dependencies in the model and the prioritizations. In other words, the experts have validated the qualitative part of the underlying theory. The quantitative part of the theory has been validated on a component level in the studies from which the theory is developed. CySeMoL's theory is drawn from the experts directly or from published empirical studies in the domain. Thus, further tests on a component level of the quantitative model's validity by experts would be redundant.

To test the validity of CySeMoL on a system level, a Turing test was performed. Turing tests are particularly useful when the answers to test cases are unknown (or costly to determine) and it cannot be assumed that a particular domain expert is correct [59]. The Turing test was designed to validate the attack paths and estimates against the criterion, i.e., that CySeMoL performs as a domain expert. Turing tests of expert systems have several advantages over other tests [59]. However, no standards have been established for how the Turing tests should be designed. The test of CySeMoL was similar to the tests described in [68] and [71]. Two pools of human experts are used: one that produces assessments of the same type as the expert system and one that rates the first pool's assessments and the expert system's assessments based on how reasonable the assessments are. The test's design is described below.

Three system architectures were presented to five domain experts experienced in penetration testing. The system architectures were depicted in a graphical format together with tables showing attributes of objects in the architecture during interviews lasting one hour. The graphical drawings and tables contained the information prescribed by CySeMoL's metamodel (cf. section 5.1). The five domain experts were asked to reason about ways that three different attack goals could be reached in the system architecture. The experts were asked to focus on the attacks with a relatively high probability of success, i.e., to disregard attacks that are unlikely to succeed. The resulting attack scenarios contained a brief description of the attack and estimates of the probability that a professional penetration tester would succeed with the included attack steps within one week. During the hour-long interviews, the experts produced one to three attack paths or solutions for each of the nine cases presented.

To limit the time required to evaluate these solutions, a subset of the five experts' solutions was used in the Turing test. One solution from each expert was randomly selected for each of the nine cases. The same principle was applied when solutions from CySeMoL were selected: one solution was randomly selected from the three solutions with the highest probability of success. Thus, solutions to nine cases from six experts were used. One of these experts was CySeMoL. To disguise the sources of the attack scenarios, the scenarios were described using the same language and abstraction level. In addition, all probabilities were rounded to the nearest percentile, which is a factor of 5 % because this resolution was used by most of the experts.

The database of 54 solutions was then presented in randomized order in a questionnaire to two domain experts. Using a five-point scale, the experts were asked to say if they agreed with the statement "this assessment is reasonable and correct". On this scale, one means that the evaluator completely disagrees with the statement. Five means that the evaluator completely agrees with the statement.

The sample size prohibits drawing reliable statistical conclusions from this test. The median score that the evaluators gave the experts and CySeMoL attack scenarios is shown in Table 2. The summary statistics indicate that the reasonableness of CySeMoL's assessments is comparable to that of the assessments of the domain experts. In mean score, CySeMoL ranked fourth in a tie with expert 2. In median score, CySeMoL ranked fifth.

Table 2. Results from the Turing test

	Evaluator 1	Evaluator 2	Mean	Median
Expert 1	[2,4,3,2,2,2,5,4,3]	[4,4,3,4,4,2,4,4,4]	3.3	4
Expert 2	[4,4,2,2,4,2,3,2,1]	[4,4,3,3,4,2,2,4,3]	2.8	3
Expert 3	[2,4,3,4,3,3,3,4,3]	[4,2,4,5,3,4,2,4,3]	3.3	3
Expert 4	[4,1,4,2,2,3,4,4,4]	[4,2,4,3,3,3,3,4,3]	3.2	3
Expert 5	[2,2,2,1,1,1,2,2,2]	[2,2,2,2,2,2,2,2,2]	1.8	2
CySeMoL	[2,2,3,1,2,2,3,3,2]	[5,5,4,3,4,2,1,4,2]	2.8	2.5
Novice 1	[2,4,3,1,2,2,2,3,2]	[2,3,2,2,2,2,2,2,2]	2.2	2
Novice 2	[1,2,4,1,2,2,2,1,1]	[3,3,3,4,2,2,3,2,2]	2.2	2
Novice 3	[4,2,2,4,4,4,3,2,1]	[2,2,2,3,3,2,2,2,1]	2.5	2

Considerable variation exists between the evaluators' scores. A potential concern is that the scoring is arbitrary, i.e., that the experts are unable to distinguish a reasonable solution from an unreasonable one. To test the discriminatory ability of the evaluators, they were requested to evaluate the solutions of three IT security novices. These novices had more than five years of experience in information technology but without a focus on security matters. The novices' solutions were elicited the same way as the experts' solutions were elicited and their solutions were presented in the same format to the evaluators. The evaluators did not know that these solutions were produced by novices.

As shown in Table 2, the novices score better individually than one expert. However, the novices receive low median and mean scores compared with the experts overall, suggesting that the evaluators can discriminate reasonable solutions from less reasonable.

6.3 Applicability and usability

To apply CySeMoL, the user must model the system architecture according to the metamodel depicted in Figure 1. However, the user is not required to input all the information in the metamodel. In particular, the user does not need to input the attributes included in the lower box of the classes (the attack steps). Thus, the user is required to model concepts such as network zone, data flow, and software installation and assign values to attributes that determine whether countermeasures such as DNSSEC and non-executable memory are functioning. However, the user is not required to ascertain whether attacks succeed. In addition, for a number of attributes, the PRM can estimate values for the attributes in the upper tile. For example, the presence of unpatched publicly known vulnerabilities in installed software can be estimated based on the product's attributes and the presence of automated patching procedures.

The usability of this tool has been assessed in [63]. Areas for improvement in graphical attractiveness and the automated support for time-demanding tasks are identified in [63]. However, users without security expertise can comprehend the concepts used to model systems with CySeMoL when the textual definition of the concepts is presented.

CySeMoL has also been evaluated with respect to usability in three case studies that analyzed the security of system architectures. The case studies focused on the following: (1) the control center and adjacent environments of one of Sweden's largest electrical power utilities, (2) the electrical substations and remote communication of one of Sweden's largest power systems, and (3) reference architectures for one of the world's most common electrical-power management systems. An excerpt from an instance model with one assessed attack path is depicted in Figure 2

The results of these three CySeMoL applications were appreciated by the system owners, and previously unknown security issues were identified in all three studies. However, the potential for improvement in data collection and the visualization of assessment results was identified. Meanwhile, data-collection support has been implemented (see [64]).

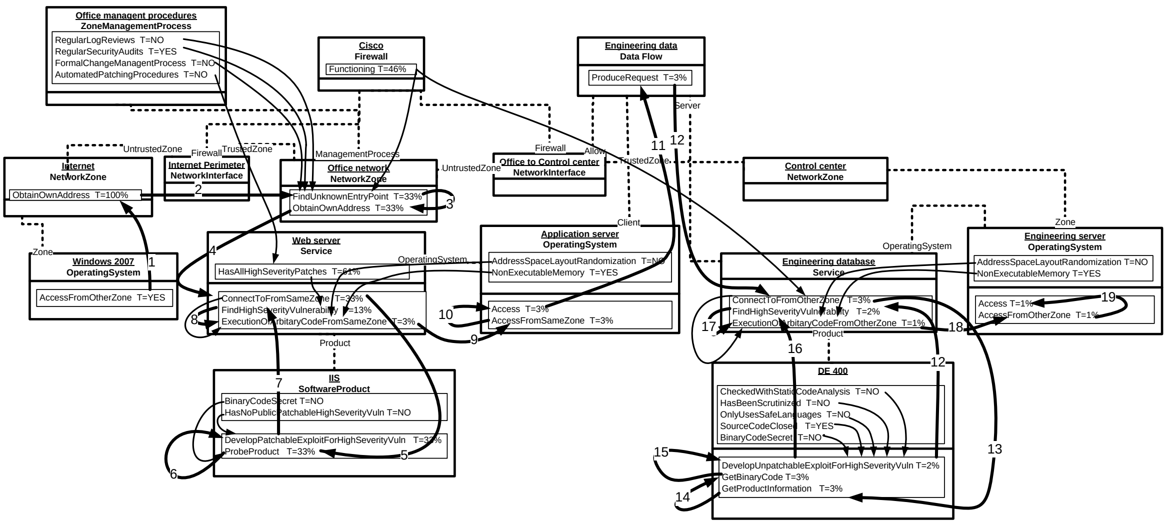


Figure 2. An excerpt from an instance model of 19-step attack path together with the probabilities that each step along this path will be reached (T=True). The links in the attack path are the enumerated bold arrows.

7 Summary and Future work

CySeMoL is a modeling language coupled to an inference engine for analyzing the security of enterprise system architectures. The inference engine produces attack paths from one attack step to another. For these attack paths, the inference engine estimates the probability that the attack can be accomplished by a professional penetration tester within one week using publicly available tools.

CySeMoL has been implemented in an existing tool [60] and validated on the component and system levels. On the component level, the theory specified in the dependencies is drawn from empirical studies in domain security and domain experts. On the system level, a Turing test suggests that the reasonableness of assessments produced by CySeMoL compares with that of a security expert and that both CySeMoL and the experts are more reasonable than security novices. These results suggest that CySeMoL would be useful where no security expert is available.

These results are promising. They suggest that assembling the body of system-security knowledge in a tool that can automate the assessments produced by experts in the field is feasible. Further work can be directed towards increasing CySeMoL's scope, refining and testing the model's accuracy, and maintaining and updating the theory.

When it comes to the scope, CySeMoL has been developed to support decision making related to the security of industrial control systems. This design focus has delimited the attacks that are covered by CySeMoL. Particularly, attacks on confidentiality are not well covered by CySeMoL because confidentiality is of lesser importance in industrial-control systems than in many other information systems. Further work is required if CySeMoL is to cover such attacks in a comprehensive manner. Effort can also be applied in modeling how attackers behave (i.e., determining which attacks attackers will attempt) and the consequences of successful attacks (i.e., to assess expected losses).

When it comes to accuracy, further tests are required to assess CySeMoL's accuracy with confidence. These tests can be on a component level and test a few probabilities or on system level and test the attack paths predicted for system architectures. Realization values can be sought in empirical tests, e.g., in conjunction with security tests or security exercises and competitions. Research can also be focused on refining the model and improving CySeMoL's accuracy. CySeMoL has been designed to produce assessments at a reasonable cost. In other words, it should not be overly costly to model a system-of-systems using CySeMoL. Work that refines the theory of CySeMoL by adding more detail to the metamodel to improve accuracy should take the cost of using these additions into consideration.

The threat environment and the countermeasures used at enterprises change over time. These changes will decrease CySeMoL's accuracy and value unless the theory is maintained and updated. As discussed in [65], some changes have a fundamental effect on the security domain. For example, when operating systems with containing new countermeasures become widely adopted. When fundamental changes occur, they are hopefully easy to identify along with the components of the theory they affect.

Other changes have limited impact on the overall threat environment or IT-landscape of enterprises. CySeMoL covers the most frequent of these changes. For instance, CySeMoL can detect the discovery of new vulnerabilities in a software product. Smaller changes that are not covered by CySeMoL can be problematic to detect and adjust the theory for. Regular reviews of the theory (e.g., annual Turing tests) will be required to identify such gradual evolutions of the threat environment and the IT landscape. In any case, the theory will require ongoing study to preserve its accuracy

8 References

[1] T. Sommestad, M. Ekstedt, and P. Johnson, "A Probabilistic Relational Model for Security Risk Analysis," *Computers & Security*, 2010.

[2] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar, "Probabilistic Relational Models," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2007, pp. 129-175.

[3] V. Verendel, "Quantified security is a weak hypothesis: a critical survey of results and assumptions," *New Security Paradigms Workshop*, 2009.

[4] IEEE/IEC, "Information technology -- Security techniques -- Information security management measurements, ISO/IEC 27004," Geneva, Switzerland, 2009.

[5] M. Swanson, N. Bartol, J. Sabato, J. Hash, and Laurie Graffo, "Security Metrics Guide for Information Technology Systems," *NIST Special Publications*, vol. 800, no. 55, 2003.

[6] B. Schneier, "Attack trees: Modeling security threats," *Dr. Dobb's Journal*, 1999.

[7] S. Bistarelli, F. Fioravanti., and P. Peretti., "Defense trees for economic evaluation of security investments," 2006, pp. 416-423.

[8] L. Piètre-Cambacédès and M. Bouissou, "Beyond Attack Trees: Dynamic Security Modeling with Boolean Logic Driven Markov Processes (BDMP)," *2010 European Dependable Computing Conference*, pp. 199-208, 2010.

[9] M. S. Lund, B. Solhaug, and K. Stolen, *Model-driven risk analysis: the CORAS approach*. Springer Verlag, 2011.

[10] H. Mouratidis, P. Giorgini, G. Manson, and I. Philp, "A natural extension of tropos methodology for modelling security," in *the Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002)*, 2002.

[11] R. Breu, F. Innerhofer-Oberperfler, and A. Yautsiukhin, "Quantitative Assessment of Enterprise Security System," *2008 Third International Conference on Availability, Reliability and Security*, pp. 921-928, Mar. 2008.

[12] H. Pardue, J. Landry, and A. Yasinsac, "A risk assessment model for voting systems using threat trees and monte carlo simulation," in *Requirements Engineering for e-Voting Systems (RE-VOTE), 2009 First International Workshop on*, 2010, pp. 55–60.

[13] H. Pardue, J. P. Landry, and A. Yasinsac, "E-Voting Risk Assessment," *International Journal of Information Security and Privacy*, vol. 5, no. 3, pp. 19-35, 2011.

[14] P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System (CVSS), Version 2.0, Forum of Incident Response and Security Teams." 2007.

[15] M. McQueen, W. Boyer, M. Flynn, and G. Beitel, "Time-to-compromise model for cyber risk reduction estimation," *Quality of Protection*, 2006.

[16] E. Johansson, "Assessment of Enterprise Information Security–How to make it Credible and efficient," KTH - The Royal Insitute of Technology, 2005.

[17] T. Heberlein, M. Bishop, E. Ceesay, M. Danforth, and CG, "A Taxonomy for Comparing Attack-Graph Approaches," *netsq.com*, pp. 1-14.

[18] S. Roschke, F. Cheng, R. Schuppenies, and C. Meinel, "Towards Unifying Vulnerability Information for Attack Graph Construction," in *Proceedings of the 12th International Conference on Information Security*, 2009, p. 233.

[19] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, 2000, pp. 307-321.

[20] O. M. Sheyner, "Scenario graphs and attack graphs," Carnegie Mellon University, 2004.

[21] R. Lippmann, "Netspa: A network security planning architecture," Massachusetts Institute of Technology, 2002.

[22] R. Lippmann et al., "Validating and restoring defense in depth using attack graphs," in *MILCOM*, 2006, p. 10 pp. -.

[23] J. Homer, K. Manhattan, X. Ou, and D. Schmidt, "A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks," Kansas, 2010.

[24] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, and K. Prole, *Advances in Topological Vulnerability Analysis*. Washington, DC: IEEE, 2009, pp. 124-129.

[25] R. P. Lippmann and L. L. C. Williams, "GARNET: a Graphical Attack graph and Reachability Network Evaluation Tool," in *Visualization for Computer Security*, K. Prole, Ed. Heidelberg-Berlin: Springer Berlin / Heidelberg, 2008, pp. 44-59.

[26] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer, "Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR," in *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, 2010, pp. 22–33.

[27] R. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *13th European Symposium on Research in Computer Security (ESORICS)*, 2008, no. 0716665, pp. 18-34.

[28] H. Holm, T. Sommestad, J. Almroth, and M. Persson, "A quantitative evaluation of vulnerability scanning," *Information Management & Computer Security*, vol. 19, no. 4, 2011.

[29] K. Stouffer, J. Falco, and K. Kent, "Guide to Industrial Control Systems (ICS) Security Recommendations of the National Institute of Standards and Technology," *Nist Special Publication*, vol. 800, no. 82, 2008.

[30] K. Ingols, M. Chu, R. Lippmann, and S. Webster, "Modeling Modern Network Attacks and Countermeasures Using Attack Graphs," in *Annual Computer Security Applications Conference*, 2009, pp. 117-126.

[31] F. . Jensen, *Bayesian Networks and Decision Graphs*. Secaucus, NJ, USA.: Springer New York, 2001.

[32] R. J. Anderson, *Security Engineering: A guide to building dependable distributed systems*. New York, NY, USA: Wiley Publishing, 2008.

[33] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, p. 39, Apr. 2004.

[34] The MITRE Corporation, "The Common Attack Pattern Enumeration and Classification," *(website)*, 2011. [Online]. Available: http://capec.mitre.org/.

[35] J. Wilander and M. Kamkar, "A comparison of publicly available tools for dynamic buffer overflow prevention," in *Proceedings of the 10th Network and Distributed System Security Symposium*, 2003, pp. 149–162.

[36] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole, "Buffer Overflows : Attacks and Defenses for the Vulnerability of the Decade," in *Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems]*, 2003, pp. 227-237.

[37] N. Frykholm, "Countermeasures against buffer overflow attacks," *RSA Tech Note*, pp. 1-9, 2000.

[38] I. Simon, "A comparative analysis of methods of defense against buffer overflow attacks," *Web address: http://www. mcs. csuhayward. edu/\~ simon/security/boflo. html*, pp. 1-16, 2001.

[39] Y. Younan, "Efficient countermeasures for software vulnerabilities due to memory management errors," Katholieke Universiteit Leuven, 2008.

[40] S. Marechal, "Advances in password cracking," *Journal in Computer Virology*, vol. 4, no. 1, pp. 73-81, 2007.

[41] M. Dell' Amico, P. Michiardi, and Y. Roudier, "Password Strength: An Empirical Analysis," *2010 Proceedings IEEE INFOCOM*, pp. 1-9, Mar. 2010.

[42] J. Cazier, "Password security: An empirical investigation into e-commerce passwords and their crack times," *Information Security Journal: A Global*, 2006.

[43] "Free Rainbow Tables," 2011. [Online]. Available: http://www.freerainbowtables.com/. [Accessed: 01-Apr-2011].

[44] J. McHugh, "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262-294, Nov. 2000.

[45] A. Ozment, "Improving vulnerability discovery models," in *Proceedings of the 2007 ACM workshop on Quality of protection*, 2007, pp. 6–11.

[46] T. Sommestad, H. Holm, and M. Ekstedt, "Effort estimates for vulnerability discovery projects," in *HICSS'12: Proceedings of the 45th Hawaii International Conference on System Sciences*, 2012.

[47] T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of remote arbitrary code execution attacks," *Information Management & Computer Security*, vol. (Accepted .

[48] T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of Denial-of-Service attacks," in *TrustCom 2011*, 2011, no. 1.

[49] T. Sommestad, H. Holm, M. Ekstedt, and N. Honeth, "Quantifying the effectiveness of intrusion detection systems in operation through domain experts," 2012.

[50] R. Cooke, "TU Delft expert judgment data base," *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May 2008.

[51] T. Sommestad, "Exploiting network configuration mistakes: practitioners self-assessed success rate," Stockholm, Sweden, 2011.

[52] T. Sommestad, M. Ekstedt, H. Holm, and M. Afzal, "Security mistakes in information system deployment projects," *Information Management and Computer Security*, vol. 19, no. 2, 2011.

[53] A. Wool, "A quantitative study of firewall configuration errors," *Computer*, pp. 62–67, 2004.

[54] J. R. Jacobs, "Measuring the Effectiveness of the USB Flash Drive as a Vector for Social Engineering Attacks on Commercial and Residential Computer Systems," Embry Riddle Aeronautical University, 2011.

[55] S. Stasiukonis, "Social engineering, the USB way," *Dark Reading*, vol. 7, 2006.

[56] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Communications of the ACM*, vol. 50, no. 10, pp. 94–100, Mar. 2007.

[57] R. Dodge and A. Ferguson, "Using Phishing for User Email Security Awareness," in *Security and Privacy in Dynamic Environments*, vol. 201, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds. Springer Boston, 2006, pp. 454-459.

[58] M. Buschle, "KTH | The Enterprise Architecture Tool," 2011. [Online]. Available: http://www.kth.se/ees/omskolan/organisation/avdelningar/ics/research/eat. [Accessed: 28-Sep-2011].

[59] R. M. O'Keefe and D. E. O'Leary, "Expert system verification and validation: a survey and tutorial," *Artificial Intelligence Review*, vol. 7, no. 1, pp. 3-42, Feb. 1993.

[60] M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A Tool for Enterprise Architecture Analysis using the PRM formalism," in *Proc. CAiSE Forum 2010*, 2010.

[61] M. J. Druzdzel, "GeNIe: A development environment for graphical decision-analytic models," in *Proceedings of the 1999 Annual Symposium of the American Medical Informatics Association (AMIA-1999)*, 1999, p. 1206.

[62] R. Agarwal, R. Kannan, and M. Tanniru, "Formal validation of a knowledge-based system using a variation of the Turing test," *Expert Systems with Applications*, vol. 6, no. 2, pp. 181-192, Apr. 1993.

[63] M. Österlind, "Validering av vektyget Enterprise Architecture Tool," Royal Institute of Technology (KTH), 2011.

[64] M. Buschle, H. Holm, T. Sommestad, M. Ekstedt, and K. Shahzad, "A Tool for automatic Enterprise Architecture modeling," in *CAISE'11 Forum*, 2011.

[65] D. Ahmad, "The Contemporary Software Security Landscape," *IEEE Security & Privacy Magazine*, vol. 5, no. 3, pp. 75-77, May 2007.