
# A Framework and Calculation Engine for Modeling and Predicting the Cyber Security of Enterprise Architectures

HANNES HOLM

Doctoral Thesis
Stockholm, Sweden 2014

TRITA EE 2014:001
ISBN 978-91-7595-005-1
ISSN 1653-5146
ISRN KTH/ICS/R--14/01--SE

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

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



Set in LaTeX by the author
Printed by Universitetsservice US AB

# Abstract

Information Technology (IT) is a cornerstone of our modern society and essential for governments' management of public services, economic growth and national security. Consequently, it is of importance that IT systems are kept in a dependable and secure state. Unfortunately, as modern IT systems typically are composed of numerous interconnected components, including personnel and processes that use or support it (often referred to as an enterprise architecture), this is not a simple endeavor. To make matters worse, there are malicious actors who seek to exploit vulnerabilities in the enterprise architecture to conduct unauthorized activity within it. Various models have been proposed by academia and industry to identify and mitigate vulnerabilities in enterprise architectures, however, so far none has provided a sufficiently comprehensive scope.

The contribution of this thesis is a modeling framework and calculation engine that can be used as support by enterprise decision makers in regard to cyber security matters, e.g., chief information security officers. In summary, the contribution can be used to model and analyze the vulnerability of enterprise architectures, and provide mitigation suggestions based on the resulting estimates. The contribution has been tested in real-world cases and has been validated on both a component level and system level; the results of these studies show that it is adequate in terms of supporting enterprise decision making.

This thesis is a composite thesis of eight papers. Paper 1 describes a method and dataset that can be used to validate the contribution described in this thesis and models similar to it. Paper 2 presents what statistical distributions that are best fit for modeling the time required to compromise computer systems. Paper 3 describes estimates on the effort required to discover novel web application vulnerabilities. Paper 4 describes estimates on the possibility of circumventing web application firewalls. Paper 5 describes a study of the time required by an attacker to obtain critical vulnerabilities and exploits for compiled software. Paper 6 presents the effectiveness of seven commonly used automated network vulnerability scanners. Paper 7 describes the ability of the signature-based intrusion detection system Snort at detecting attacks that are more novel, or older than its rule set. Finally, paper 8 describes a tool that can be used to estimate the vulnerability of enterprise architectures; this tool is founded upon the results presented in papers 1-7.

**Keywords:** Computer security, security metrics, vulnerability assessment, attack graphs, risk management, architecture modeling, Enterprise Architecture

# Sammanfattning

Informationsteknik (IT) är en grundsten i vårt moderna samhälle och grundläggande för staters hantering av samhällstjänster, ekonomisk tillväxt och nationell säkerhet. Det är därför av vikt att IT-system hålls i ett tillförlitligt och säkert tillstånd. Då moderna IT-system vanligen består av en mångfald av olika integrerade komponenter, inklusive människor och processer som nyttjar eller stödjer systemet (ofta benämnd organisationsövergripande arkitektur, eller enterprise architecture), är detta tyvärr ingen enkel uppgift. För att förvärra det hela så finns det även illvilliga aktörer som ämnar utnyttja sårbarheter i den organisationsövergripande arkitekturen för att utföra obehörig aktivitet inom den. Olika modeller har föreslagits av den akademiska världen och näringslivet för att identifiera samt behandla sårbarheter i organisationsövergripande arkitekturer, men det finns ännu ingen modell som är tillräckligt omfattande.

Bidraget presenterat i denna avhandling är ett modelleringsramverk och en beräkningsmotor som kan användas som stöd av organisatoriska beslutsfattare med avseende på säkerhetsärenden. Sammanfattningsvis kan bidraget användas för att modellera och analysera sårbarheten av organisationsövergripande arkitekturer, samt ge förbättringsförslag baserat på dess uppskattningar. Bidraget har testats i fallstudier och validerats på både komponentnivå och systemnivå; resultaten från dessa studier visar att det är lämpligt för att stödja organisatoriskt beslutsfattande.

Avhandlingen är en sammanläggningsavhandling med åtta artiklar. Artikel 1 beskriver en metod och ett dataset som kan användas för att validera avhandlingens bidrag och andra modeller likt detta. Artikel 2 presenterar vilka statistiska fördelningar som är bäst lämpade för att beskriva tiden som krävs för att kompromettera en dator. Artikel 3 beskriver uppskattningar av tiden som krävs för att upptäcka nya sårbarheter i webbapplikationer. Artikel 4 beskriver uppskattningar för möjligheten att kringgå webbapplikationsbrandväggar. Artikel 5 beskriver en studie av den tid som krävs för att en angripare skall kunna anskaffa kritiska sårbarheter och program för att utnyttja dessa för kompilerad programvara. Artikel 6 presenterar effektiviteten av sju vanligt nyttjade verktyg som används för att automatiskt identifiera sårbarheter i nätverk. Artikel 7 beskriver förmågan av det signaturbaserade intrångsdetekteringssystemet Snort att upptäcka attacker som är nyare, eller äldre, än dess regeluppsättning. Slutligen beskriver artikel 8 ett verktyg som kan användas för att uppskatta sårbarheten av organisationsövergripande arkitekturer; grunden för detta verktyg är de resultat som presenteras i artikel 1-7.

**Nyckelord:** Cybersäkerhet, säkerhetsmetriker, sårbarhetsanalys, attackgrafer, riskhantering, arkitekturmodellering, organisationsövergripande arkitektur

# Acknowledgments

While it is difficult to produce an exhaustive list of all those who have contributed to this thesis, or supported me during this journey, it is clear that it would not have been possible without your aid.

Mathias Ekstedt, Robert Lagerström and Göran Ericsson provided valuable input in their role of supervisors. Judith Westerlund and Annica Johannesson deserve many thanks for making the administrative aspects of my academic life so much easier.

My colleagues at both ICS and abroad (in particular, the Argus group) have all contributed to this thesis with interesting discussions and for having created an atmosphere that made work both rewarding and joyful. It has sincerely been a pleasure working with you. I am particularly grateful to my paper co-authors, especially Teodor Sommestad who served almost as a fourth supervisor during my first year as a Ph.D. student. Another special thanks goes to Kun Zhu and Markus Buschle for all pain spent at KTH-hallen.

A number of people within the industry have aided the development of this thesis with unvaluable practical insight and data; it could definitely not have been done without you.

Finally, I would like to thank Anna, my family - Helen, Stefan and Jesper - and my friends - in particular, Ricard and Markus. Thank you for your support, and especially, for tolerating my endless monologues on IT security!

Stockholm, January 2014
Hannes Holm

# Papers

## Papers included in the thesis

[1] Hannes Holm, Mathias Ekstedt, and Dennis Andersson,"Empirical Analysis of System-Level Vulnerability Metrics through Actual Attacks", in *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 6, pp. 825–837, Nov.-Dec. 2012, DOI: 10.1109/TDSC.2012.66.

[2] Hannes Holm, "A Large-Scale Study of the Time Required to Compromise a Computer System", in *Dependable and Secure Computing, IEEE Transactions on*, vol. 10, no. 6, Nov.-Dec. 2013, DOI: 10.1109/TDSC.2013.21.

[3] Hannes Holm, Mathias Ekstedt, and Teodor Sommestad, "Effort Estimates for Web Application Vulnerability Discovery", in *46th Hawaii International Conference on Systems Sciences (HICSS)*, Maui, Hawaii, January 7-10 2013, pp. 5029–5038.

[4] Hannes Holm, and Mathias Ekstedt, "Estimates on the Effectiveness of Web Application Firewalls Against Targeted Attacks", in *Information Management and Computer Security*, vol. 21, no. 4, pp. 250–265, Nov. 2013, DOI: 10.1108/IMCS-11-2012-0064.

[5] Hannes Holm, Matus Korman, and Mathias Ekstedt, "A Bayesian Model for Likelihood Estimations of Acquirement of Critical Software Vulnerabilities and Exploits", submitted manuscript.

[6] Hannes Holm, "Performance of Automated Network Vulnerability Scanning at Remediating Security Issues", in *Computers & Security*, vol. 31, no. 2, pp. 164–175, Mar. 2012, DOI: 10.1016/j.cose.2011.12.014.

[7] Hannes Holm, "Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter?", in *47th Hawaii International Conference on Systems Sciences (HICSS)*, Big Island, Hawaii, January 6-9 2014, pp. 4895–4904.

[8] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt, "$P^2$CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language", submitted manuscript.

### Author contributions

In all papers, Hannes Holm has been the leading researcher and primary author.

In [1], the general research concept is due to Holm, the data collection to Holm and Andersson, the analysis to Holm and Ekstedt, and the authoring mostly to Holm.

[2, 6, 7] were authored solely by Holm.

In [3], the general research concept is due to Holm and Ekstedt, the survey instrument to Holm, Sommestad and Ekstedt, the data collection and analysis to Holm, and the authoring mostly to Holm and Sommestad.

In [4], the general research concept, survey instrument and authoring are due to Holm and Ekstedt, and the data collection and analysis to Holm.

In [5], the general research concept is due to Holm and Ekstedt, the survey instrument, scripts, data collection and analysis to Holm, and the authoring to Holm with assistance from Korman and Ekstedt.

In [8], the general research concept is mostly due to Holm and Ekstedt, the $P^2AMF$ implementation to Holm, Shahzad, Buschle and Ekstedt, the case studies to Holm with assistance from Ekstedt, the validation to Holm, and the authoring mostly to Holm, Buschle and Ekstedt.

## Related papers not included in the thesis

[9] Robert Lagerström, Liv Marcks von Würtemberg, Hannes Holm, and Oscar Luczak, "Identifying factors affecting software development cost," in *Fourth International Workshop on Software Quality and Maintainability (SQM)*, Mar. 2010.

[10] Mark Jensen, Cumhur Sel, Ulrik Franke, Hannes Holm, and Lars Nordstrom, "Availability of a SCADA/OMS/DMS system – A case study," in *Innovative Smart Grid Technologies Conference Europe (ISGT Europe)*, pp. 1–8, Oct. 2010.

[11] Per Närman, Hannes Holm, Pontus Johnson, Johan König, Moustafa Chenine, and Mathias Ekstedt, "Data accuracy assessment using enterprise architecture," in *Enterprise Information Systems*, vol. 5, no. 1, pp. 37–58, 2011, DOI: 10.1080/17517575.2010.507878.

[12] Hannes Holm, Teodor Sommestad, Ulrik Franke, and Mathias Ekstedt, "Expert assessment on the probability of successful remote code execution attacks," in *8th International Workshop on Security in Information Systems – WOSIS 2011*, Jun. 2011.

[13] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm, "Security mistakes in information system deployment projects," in *Information Management & Computer Security*, vol. 19, no. 2, pp. 80–94, 2011, DOI: 10.1108/09685221111143033.

[14] Teodor Sommestad, Hannes Holm, and Mathias Ekstedt, "Estimates of success rates of Denial-of-Service attacks," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pp. 21–28, 2011.

[15] Waldo Rocha Flores, Teodor Sommestad, and Hannes Holm, "Assessing future value of investments in IT governance control objectives – an expert-based evaluation," in *Electronic Journal of Information Systems Evaluation*, vol. 14, no. 2, pp. 216–227, 2011.

[16] Waldo Rocha Flores, Teodor Sommestad, and Hannes Holm, "Assessing future value of investments in IT governance control objectives – an expert-based evaluation," in *5th International Conference on Information Management and Engineering (ICIME)*, Apr. 2011.

[17] Markus Buschle, Hannes Holm, Teodor Sommestad, and Mathias Ekstedt, "Architectural analysis using automated vulnerability assessment," in *23rd International Conference on Advanced Information Systems Engineering (CAiSE'11)*, Jun. 2011.

[18] Dennis Andersson, Magdalena Granåsen, Thomas Sundmark, Hannes Holm, and Jonas Hallberg, "Analysis of a Cyber Defense Exercise using Exploratory Sequential Data Analysis," in *Proceedings of the 16th International Command and Control Research and Technology Symposium (ICCRTS)*, Jun. 2011.

[19] Hannes Holm, Teodor Sommestad, Jonas Almroth, and Mats Persson, "A quantitative evaluation of vulnerability scanning," in *Information Management & Computer Security*, vol. 19, no. 4, pp. 231–247, 2011, DOI: 10.1108/09685221111173058.

[20] Dennis Andersson, Magdalena Granåsen, Thomas Sundmark, Hannes Holm, and Jonas Hallberg, "Exploratory Sequential Data Analysis of a Cyber Defence Exercise," in *Proceedings of the International Defense and Homeland Security Simulation Workshop (DHSS)*, Sep. 2011.

[21] Teodor Sommestad, Hannes Holm, and Mathias Ekstedt, "Threats and vulnerabilities, final report," *Stockholm, Sweden: Report of The VIKING project*, 2011.

[22] Hannes Holm, Teodor Sommestad, Mathias Ekstedt, "Vulnerability assessment of SCADA systems," *Stockholm, Sweden: Report of The VIKING project*, 2011.

[23] Teodor Sommestad, Hannes Holm, and Mathias Ekstedt, "Effort estimates for vulnerability discovery projects," in *45th Hawaii International Conference on System Science (HICSS)*, pp. 5564–5573, Jan. 2012.

[24] Hannes Holm, Teodor Sommestad, Ulrik Franke, and Mathias Ekstedt, "Success rate of remote code execution attacks – expert assessments and observations," *Journal of Universal Computer Science*, vol. 18, no. 6, pp. 732–749, Mar. 2012.

[25] Teodor Sommestad, Hannes Holm, and Mathias Ekstedt, "Estimates of success rates of remote arbitrary code execution attacks," in *Information Management & Computer Security*, vol. 20, no. 2, pp. 107–122, 2012, DOI: 10.1108/09685221211235625.

[26] Markus Buschle, Hannes Holm, Teodor Sommestad, and Mathias Ekstedt, "A Tool for Automatic Enterprise Architecture Modeling," in *IS Olympics: Information Systems in a Diverse World*, vol. 107, pp. 1–15, 2012.

[27] Robert Lagerström, Liv Marcks von Würtemberg, Hannes Holm, and Oscar Luczak, "Identifying factors affecting software development cost and productivity," in *Software quality journal*, vol. 20, no. 2, pp. 395–417, Jun. 2012, DOI: 10.1007/s11219-011-9137-8.

[28] Hannes Holm, Markus Buschle, Robert Lagerström, and Mathias Ekstedt, "Automated data collection for enterprise architecture models," in *Software & Systems Modeling*, pp. 1–17, Jun. 2012, DOI: 10.1007/s10270-012-0252-1.

[29] Per Närman, Hannes Holm, David Höök, Nicholas Honeth, and Pontus Johnson, "Using enterprise architecture and technology adoption models to predict application usage," in *Journal of Systems and Software*, vol. 85, no. 8, pp. 1953–1967, Aug. 2012, DOI: 10.1016/j.jss.2012.02.035.

[30] Hannes Holm, "Baltic Cyber Shield: Research from a red team versus blue team exercise," in *PenTest magazine*, vol. 2, no. 5, pp. 80–86, 2012.

[31] Hannes Holm and Mathias Ekstedt, "A metamodel for web application injection attacks and countermeasures," in *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation (TEAR)*, pp. 198–217, Oct. 2012.

[32] Per Närman, Hannes Holm, Mathias Ekstedt, and Nicholas Honeth, "Using enterprise architecture analysis and interview data to estimate service response time," in *The Journal of Strategic Information Systems*, vol. 22, no. 1, pp. 70–85, Mar. 2013, DOI: 10.1016/j.jsis.2012.10.002.

[33] Hannes Holm, Teodor Sommestad, and Mathias Ekstedt, "CySeMoL: A tool for cyber security analysis of enterprises," in *CIRED*, Jun. 2013.

[34] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm, "The Cyber Security Modeling Language – A Tool for Vulnerability Assessments of Enterprise Architectures," in *Systems Journal, IEEE*, vol. 7, no. 3, pp. 363–373, Sept. 2013, DOI: 10.1109/JSYST.2012.2221853.

[35] Waldo Rocha Flores, Hannes Holm, Gustav Svensson, and Göran Ericsson, "Using Phishing Experiments and Scenario-based Surveys to Understand Security Behaviours in Practice," in *7th International Symposium on Human Aspects of Information Security and Assurance (HAISA)*, May 2013.

[36] Hannes Holm, Teodor Sommestad, Mathias Ekstedt, and Nicholas Honeth, "Indicators of expert judgement and their significance: an empirical investigation in the area of cyber security," in *Expert Systems*, 2013, DOI: 10.1111/exsy.12039.

[37] Hannes Holm, Waldo Rocha Flores, and Göran Ericsson, "Cyber Security for a Smart Grid – What About Phishing?," in *4th European Innovative Smart Grid Technologies (ISGT) Conference*, Oct. 2013.

[38] Hannes Holm, Mathias Ekstedt, Teodor Sommestad, and Matus Korman, "A Manual for the Cyber Security Modeling Language," *Stockholm, Sweden: Report from ICS KTH*, Nov. 2013.

[39] Waldo Rocha Flores, Hannes Holm, Gustav Svensson, and Göran Ericsson, "Using Phishing Experiments and Scenario-based Surveys to Understand Security Behaviours in Practice," in *Information Management & Computer Security*, To be available.

[40] Teodor Sommestad, Hannes Holm, Mathias Ekstedt, and Nicholas Honeth, "Quantifying the effectiveness of intrusion detection system in different operational environments through domain experts," *Journal of Information System Security*, To be available.

# Table of contents

# Part I

# Introduction

# Chapter 1

# Introduction

## 1.1 Outline of the thesis

This thesis consists of two parts. The first part provides an overview of the second part that contains the research papers which constitutes the core of this thesis. This first part presents a motivation behind the thesis, related work, the results, and the employed research design. The second part consists of eight papers that have either been published in (papers 1, 2, 4, and 6), or submitted to (papers 5, 8) peer-reviewed academic journals, or published in the proceedings of peer-reviewed academic conferences (paper 3, 7).

## 1.2 Background

Information Technology (IT) is essential to businesses as it is used to support most, if not all, organizational functions. Dependable IT systems are thus essential as their failure generally has a large impact on the services provided by an enterprise. This is especially important if the IT system in question controls a physical process that enables a widely used service, such as the power grid [118], where a failure can provide significant consequences for the society - as demonstrated by, for example, the North American blackout during 2003 [14].

Here, a service *failure* is an event that occurs when a delivered service deviates from the correct service. A failure origins from one or more *errors*; parts of a systems total state that may lead to failures. Errors are produced by active *faults*. Faults stem from failures of hardware, software, humans, or the physical world with its natural phenomena [17].

As an IT architecture typically is made up of a complex "cobweb" of interconnected IT, personnel and processes [13], often referred to as an *enterprise architecture* [103], an error in one small component - that is insignificant on its own - can have radical effects on the dependability of the overall system-of-systems [17, 137]. Consequently, it is a difficult endeavor to achieve dependable IT and various models have been proposed to measure and improve dependability [128].

To make matters worse, some active faults can enable motivated actors to conduct unauthorized activity in the enterprise architecture. If this is the case, the fault can be considered a *security vulnerability* to the dependable and secure state of the enterprise architecture [17, 22, 70].

It is certainly in the interest of enterprise decision makers to estimate where security vulnerabilities currently exist, where they might occur, and what consequences they have if successfully exploited. Unfortunately, to estimate the security vulnerability of modern enterprise architectures, an enormous amount of factors need to be considered. It is not

enough to gather information about every single known vulnerability in the architecture; there is also a need to understand, in particular, how these relate and where novel vulnerabilities might occur [181]. For instance, a critical client-side kernel vulnerability in a web server might be rather insignificant on its own as the server is not used as a client. However, this vulnerability would be problematic if the attacker is able to exploit some other vulnerability to reach it.

Decisions on how to measure and improve enterprise IT security are ultimately made by some type of enterprise decision maker, often the Chief Information Security Officer (CISO) [23]. Goodyear et al. [67] study the skills and responsibilities of the CISO, and finds that non-technical skills are in greater demand than technical skills:

> "While technical education remains important, the CISO role has grown far beyond technical management of cybersecurity tools. States should modernize the philosophical approach to cybersecurity management. At the core of effective CISO skills and competencies is a philosophy that cybersecurity problem solving is more than an exercise in technical proficiency. State CISOs themselves identify non-technical skills as particularly important, including collaboration/conflict management, communication skills, and political skills."

Consequently, a cybersecurity enterprise decision maker cannot be expected to have a deep understanding of IT security vulnerabilities and their dependencies; rather, they should be expected to have a basic understanding of their enterprise architecture and the losses incurred if assets are compromised [159].

Due to this knowledge gap, enterprise decision makers typically consult experts in order to estimate the IT security of their architectures. While consulting experts certainly is valuable, resulting estimates come with three significant delimitations: they are only valid for 1) the time that they were carried out, 2) the parts of the enterprise architecture that were studied by the expert, and 3) the competence of the consulted expert. These delimitations are especially problematic given the dynamic nature of enterprise architectures and the lack of resources available for analyses. Another means of managing cybersecurity, often tightly coupled with expert investigations, is to implement the guidelines prescribed by standards or frameworks such as the ISO/IEC 27000 series [94] or the Common Criteria [32]. However, these guidelines are by design very general and do not provide any readily available means to measure and improve security; rather, they aid with the development of metrics that can be used to measure and improve security. As a consequence, application of them can be vague and troublesome. For instance, Ross Andersson [12] reports that:

> "In none of the half-dozen or so affected cases I've been involved in has the Common Criteria approach proved satisfactory."

As there is no "silver bullet", and IT security is a complex issue, a sometimes employed method is what Bruce Schneier refer to as *Security Theater* [142]:

> "Security theater refers to security measures that make people feel more secure without doing anything to actually improve their security."

In other words, to invest in some IT security policy, education or tool to at least make it *seem* as if the topic is adequately managed. While a perception of security can have its merits [134], implementation of security measures without proper evidence can naturally also be problematic - its cost is real, but its benefits are generally negligible.

Enterprise decision makers are thus in need of tools that can help estimate the cybersecurity of enterprise architectures in a both useful and easy-to-understand fashion. There are various research efforts that have been conducted for this purpose (see Chapter 2 for an overview). Of these, the attack graph approach is often considered the best suited method for estimating the security of enterprise architectures. Attack graphs involve usage of formal reasoning and graphical modeling to present possible attack paths corresponding to a certain architecture. According to a recent survey by [101], there are more than 30 different types of attack graph approaches. However, while there is a myriad of methodologies, there is not yet any tool that provides satisfactory analysis of the security of an enterprise architecture; they are either limited in terms of assumptions, scope or too effort-demanding to employ.

## 1.3 Purpose

The purpose of this research is to help enterprise decision makers analyze the cybersecurity, or vulnerability, of their enterprise architectures in a meaningful and understandable way. More specifically, this involves creation of a model that can be used to estimate the vulnerability of an enterprise architecture. This model should allow modifications of depicted architectures to enable estimates on the vulnerability of alternative configurations. Moreover, it should not be overly costly to employ, or require any major security expertise to use. Finally, it should be *reasonably correct*, i.e., there should exist a reasonable trade-off between data collection costs and quality of estimates. The gain of allowing some uncertainty in the result is that the data collection effort can be kept at an acceptable level, thus enabling practical usage of the model.

# Chapter 2

# Related work

The contribution of this thesis is a tool that can be used to measure the vulnerability of enterprise architectures. Thus, the related works described in this chapter concern the subject of security measurement. To understand the purpose of measurement, we paraphrase Lord Kelvin:

> "To measure is to know."
>
> "If you can not measure it, you can not improve it."
>
> "In physical science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it. I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be."

A *measurement* is the outcome of an event described according to some method. Typically, we measure to enable prediction and control [16]. A *metric* assigns measurements onto a scale in order to correctly represent some phenomenon [19]. A *security metric* concerns a metric used to correctly represent some security attribute of a system under consideration [16]. A *security model* provides a formal representation (e.g., a set of equations) that corresponds to some security attribute [176].

On an overall level, a security model or metric can be either qualitative or quantitative [139, 178]. While qualitative models such as the ISO/IEC 27004, Common Criteria, OCTAVE [3] and CORAS [45] often are used in practice, they suffer from being vague, and ultimately subjective [178]. The contribution described in this thesis is a quantitative model that allows modeling assets of enterprise architectures, and then calculating the likelihood of different attacks being successful against these assets. Consequently, this chapter delimits from presenting qualitative research and focuses on describing quantitative efforts. As the contribution computes vulnerability estimates based on measurements describing different security phenomena, Section 2.1 presents research that provide security measurements; an overview of relevant security models and metrics can be found in Section 2.2. As the contribution presented in this thesis is an attack graph approach [110], Section 2.3 is devoted to presenting related such models. Finally, the contribution is based on an existing attack

graph model, the Cyber Security Modeling Language (CySeMoL). This model is presented in Section 2.4.

## 2.1 Quantitative security measurements

Some research focus on measuring the effectiveness of different defense mechanisms under certain circumstances. These efforts can have merit for enterprise decision makers. They could also serve as input to metrication frameworks such as CORAS, OCTAVE, ISO/IEC 27004, or the contribution of this thesis.

There is a vast variety of attacks and defense mechanisms available. Exploiting an individuals' lack of security awareness to make the individual comply with a malicious request (i.e., social engineering [83]), and exploiting a vulnerable memory buffer (e.g., a buffer overflow [131]) to inject arbitrary code in a process, are examples of two radically different types of attacks that both need to be considered by an enterprise decision maker. Similarly, there are various types of defenses that aim to mitigate the same class of vulnerability. For instance, arbitrary code execution through a buffer overflow can be prevented by, e.g., application sandboxing [98], process address-space randomization [146], network- and host-based intrusion detection systems [66], proper coding practices [145], and type-safe Application Programming Interfaces (APIs) or dialects [90]. This section presents significant research that provide measurements on topics related to the scope of the contribution of this thesis (see Section 3.3).

Memory corruption attacks is a common and severe type of attack that involves redirecting the control-flow of an application to code controlled by the attacker (buffer overflow is an example from this category). Effectiveness of defenses against memory corruption attacks have been studied by, e.g., [146, 183]. Shacham et al. [146] studied Address Space Layout Randomization (ASLR), a cryptographic defense used to randomize the address space of applications in order to increase the effort by attackers to predict sought addresses. Wilander et al. [183] devised a tool for automated tests of defenses against buffer overflow attacks and used this to estimate the effectiveness of several defenses, including non-executable memory in Ubuntu 9.10. The tool first compiles a vulnerable software with the designated defenses, and then executes automated buffer overflow attacks of different types against this software. Effectiveness of a defense is given by the number of successful attacks compared to the number of unsuccessful attacks.

Detection rate of intrusion detection systems (IDSs) has been studied on previous occasions, e.g., [68, 112, 111]. The Lincoln Laboratory at MIT examined the effectiveness of 18 IDSs during 1998 and 1999, and found that the best systems detected between 63% to 93% of the tested known attacks and approximately 50% of the tested novel attacks [112, 111]. Hadžiosmanovic et al. [68] studied the effectiveness of four anomaly-based IDS at detecting various attacks (e.g., those used by [112, 111]). The authors found that the mechanisms could not provide both high detection and low false positive rates in presence of data with high variability.

The effectiveness of automated discovery of web application vulnerabilities, and detection of attacks against these, have been studied previously [15, 50, 55, 57, 168]. Fonseca et al. [57] estimated the effectiveness of different vulnerability scanners at identifying SQL and cross site scripting vulnerabilities (similar studies are presented in [15, 50]). Suto [168] examined the effectiveness of eight commercial web application firewalls. In a similar study, Elia et al. [55] examined the effectiveness of five SQL injection detection tools that operate at an application-, database-, or network-level.

Surveys of software vulnerabilities and exploits are frequently conducted by researchers and practitioners to provide knowledge on their properties (e.g., [20, 58, 60, 133, 143, 147]). Shahzad et al. [147] performed an exploratory study of archival vulnerability data spanning over 23 years, investigating aspects related to the life cycle of vulnerabilities (e.g., vulnerability disclosure and exploit release dates). Ozment and Schechter [133] examined the code base of the OpenBSD operating system to determine whether its security is increasing over time. Bilge and Dumitras [20] studied the time until discovery of zero-day attacks by observing when currently known malicious software first appeared on systems. Frei et al. [60] studied the zero-day patching efficiency of Microsoft and Apple. Fonseca et al. [58] studied 312 web application exploits and found that SQL injection and Remote File Inclusion accounted for for almost 90% of exploits analyzed. Scholte et al. [143] conducted a study of the sophistication level of over 2600 web application vulnerabilities and found that the complexity of the attacks have not changed significantly, and that many web problems still were simple in nature. By surveying known software vulnerabilities, many researchers hope to derive conclusions regarding novel software vulnerabilities that are unknown to the public community, also called zero-days' [108]. These are attractive to attackers as their exploitation cannot be prevented by applying software updates, and as a consequence, the price of these vulnerabilities can reach up to $250,000 [122].

A problem with these studies is however that they often have practical delimitations that limit their usefulness to enterprise decision making. To exemplify this problem, we consider [146], which studied the effectiveness of ASLR for a software using `fork()` on a 32-bit system. If the tested software would not have employed `fork()` (spawning a new process each time a crash occur from to a failed exploit), or the system had been 64-bit, it would likely have been radically more difficult to bypass ASLR. Furthermore, the effectiveness of ASLR is often regarded as limited in the absence of non-executable memory pages [189] due to the many means of reducing the entropy of the address-space of applications (e.g., through heap spray [136]). Also, it can be enough that a single attacker-available executable module is not compiled with ASLR to completely circumvent it (e.g., `msvcr71.dll` allowed universal bypass of ASLR and non-executable memory pages for any version of Windows during 2011 [182]). Thus, the results found by [146] might not be an accurate representation of the actual difficulty required to bypass ASLR in a non-laboratory environment. Another example is the IDS studies by [112, 111], where significant shortcomings were identified by [114, 119] - shortcomings that delimit the usefulness of these results to a real-world scenario (e.g., regarding the chosen attacks). A third example is [183], where it is unknown how common the different tested buffer overflow attack types are in practice.

One of the contributions of this thesis is to provide quantitative data on significant cybersecurity phenomena, on a level that is useful for enterprise decision making - these studies are presented in papers 3-7.

## 2.2 Quantitative security models and metrics

There is a large variety of quantitative security models and metrics that can be used as aid by enterprise decision makers. This section provides an overview of these approaches.

The arguably most well established security models concern measuring the algorithmic strength of cryptographic solutions [86]. These traditionally approximate the costs that would be incurred by an adversary to break the system by buying the latest (cheapest) tools and using the best known techniques [140]. A practical example of this concerns the RSA algorithm, which previously offered rewards of up to $200,000 for breaking the security of the keys used by its cryptosystem [138]. RSA-768, the strongest variant that was factored

during the challenge, had a reward of $50,000 and took almost 2000 2.2GHz-Opteron-CPU years (just short of 3 years of calendar time) [99]. A problem with these models is however that many cryptosystems are not broken by weaknesses in their algorithms, but by weaknesses that originate from their implementation [141].

A number of models focus on estimating the investment opportunity aspects of cybersecurity. Cavusoglu et al. [30] propose a model based on Return on Security Investment (ROSI) [165], which considers the expected trade-off by the attacker, the cost of security detection mechanisms, and expected loss due to undetected intrusions. In [100], the author presents a method for estimating costs of malware incidents and simulates the cost for 104 incidents on 10 hosts. Lelarge [104] propose a cost estimation model where strategic agents are interconnected on graphs on which malware epidemics occur. Dornseif and May [49] model the cost and benefits of a specific security tool, Honeynets. Various similar models exist, for instance, [33, 87, 148, 171].

Another group of models focus on measuring individuals' security awareness and willingness to follow organizational security policies, and how different factors affect awareness and compliance [155, 170]. In this category of models, one can find, for instance, [26, 42, 47, 149, 175, 187]. Similar models instead focus on the effectiveness of technical security mechanisms that have the purpose to increase user security awareness (e.g., the "lock" and coloring in modern web browsers to denote presence of SSL) [46, 54, 188]. A problem with security awareness related studies is however that their results often point in different directions [157]. For example, [42] found a Pearson correlation coefficient of 0.82 between subjective norms and intention to follow security policies; in a different study [109] this correlation was found to be $-0.04$.

Other models, often denoted as code metrics, focus on measuring source code characteristics thought to correlate with the occurrence of software vulnerabilities. Some of these efforts apply existing metrics originally developed with the purpose to estimate some general notion of software quality. For instance, Shin et al. [154] test whether 28 existing metrics related to complexity, code churn and developer activity correlate with occurence of software vulnerabilities within Mozilla Firefox and Red Hat Enterprise distributions. Their results indicate that 24 out of the 28 metrics are discriminative of vulnerabilities. A problem with applying such models is however that they were not developed with a focus on security. Furthermore, many of these metrics, and studies of them, have been heavily criticized [41, 56]. For instance, McCabe's cyclomatic complexity metric, which is extensively used by both researchers and practitioners (e.g., it is available in Microsoft's Visual Studio platform), received critique already during the 1980's [151].

When new models are proposed, there is typically not sufficient evidence to support them. For example, Chowdhury et al. [34] propose (among other things) a metric they denote as "stall ratio". This metric involves measuring the occurrence of statements that do not contribute to the overall progress of a program (e.g., $a = a + 0$). However, *why* this is presumed to be correlated with occurrence of software vulnerabilities is not explored. Similarly, Haller et al. [69] propose ten metrics that are presumed to correlate with occurrence of buffer errors. The authors couple each metric to a number of points (from 0 - 500) that they believe reflect the metric's relative significance. However, the motivation behind these metrics and their corresponding points is only very briefly discussed.

A group of models, often referred to as Vulnerability Discovery Models (VDMs) [4], have been proposed for predicting the occurrence of software vulnerabilities. Alhazmi et al. [5, 6, 8, 7] propose an S-shaped VDM to predict occurrence of vulnerabilities over time. The S-shape is chosen as the authors believe that vulnerability discovery begins with a long learning phase, when software testers learn how it functions. This phase is then followed by a linear accumulation phase, when testers are familiar with it and many vulnerabilities are

discovered. In the third phase the software has been replaced by newer variants, leading to a smaller user-base and consequently less interest from software testers. Similar models are proposed by [91, 97, 185].

A recent study of six significant VDMs for 17 versions of Firefox, Chrome and Internet Explorer however indicate that fit generally is poor for all tested models [127]. The authors believe that this is due to irregular vulnerability disclosures, for example, when the code base of a software is inherited in later releases.

An approach with similar purpose as VDMs, but using a different methodology, is the attack surface metric by Howard et al. [79], and Manadhata and Wing [115]. This type of metric estimates the vulnerability of software based on its attacker-accessible methods and corresponding privilege levels (here, *method* refers to the software engineering meaning of the word). Attack surface has been implemented for Microsoft Windows [172].

A wide variety of models and metrics estimate the vulnerability of systems in operation. These are related to VDMs and attack surfaces in the sense that they all concern some notion of "risk" [9]; however, they differ in the regard that they relate vulnerabilities existing in different systems in order to provide overall security estimates for systems in operation. Of these, the most common approach concerns attack graphs - these are described in Section 2.3. A few noteworthy non-attack graph based models are described next. Boyer and McQueen [24] propose a set of security metrics that can be used by operators of industrial control systems. Ahmed et. al [1] describe a model that estimates risk based on the aggregated current and historical vulnerability of operational software. Houmb et al. [78] use Bayesian belief networks to estimate the vulnerability of systems in operation. Marconato et al. [116] use stochastic activity networks, vulnerability archival data, and theory on attacker and administrator behavior to model the vulnerability of systems in operation. Alves-Foss and Salvador [11] propose a metric that is based on system characteristics, potentially neglectful acts, and potentially malevolent acts.

There are also models that focus on the architectural aspects of cyber security analyses. These models are similar to VDMs, attack surfaces and operational models in the sense that they also concern an evaluation of "risk". However, they differ in the regard that they spend particular focus on how different objects should be visually depicted and connected. Some architecture models focus on risks that might arise during software development; a significant example in this category is UMLsec [96], which extends the Unified Modeling Language (UML) with the ability to consider security issues. Other architeture models concern modeling the overall security for enterprise systems in operation. An example from this category is the model by Breu et al. [25], which provides a metamodel that concerns security issues in three layers: the business layer (business roles, activities and information), the application layer (applications used by business activities), and the technical layer (software and hardware that enables application services). A user of this model would depict and connect architecture objects in these layers that are relevant to the enterprise in question, and then quantify the likelihood of threats occurring against these objects.

Architecture models are useful in the sense that they allow enterprise decision makers to depict objects they can understand and relate to, and analysis results that are easy to comprehend. However, a problem with current architecture models is that they require the user to manually determine which objects that are reasonable to include in the model, and the risk (i.e., numbers) associated with each object. For instance, [25] does not specify what actual objects (e.g., regarding granularity) that are applicable to the application layer, or how likely different attacks are to succeed against these objects. These are both abstract and difficult tasks to manage for enterprise decision makers with limited resources available for modeling and analyses. Consequently, current architectural models suffer from the same issues as qualitative models such as the Common Criteria; i.e., they are vague, and thus

subjective [178]. The contribution described in this thesis is an architecture model that provides a comprehensive scope of objects, with less room for misinterpretation. Furthermore, it is already populated with quantitative data on the likelihood of different threats being fulfilled for these objects.

In conjunction to modeling the occurrence of software vulnerabilities, it is important to also estimate the severity of these vulnerabilities. The current standard for this purpose is the Common Vulnerability Scoring System (CVSS) [121], suite of metrics used to quantify the severity of IT vulnerabilities. The CVSS Base score, the only metric that each CVSS scored vulnerability provides measurements for, is scored on a scale from 1 (least severe) - 10 (most severe). For instance, CVE-2013-3375 (a cross-site scripting vulnerability) has a CVSS score of 4.3, and CVE-2012-6569 (a buffer overflow vulnerability) has a score of 9.3. While discussion has been raised towards the validity of the CVSS [65, 78, 113, 179], it is widely adopted by both practitioners and researchers. For example, automated vulnerability scanners such as Nessus [169] typically describe the criticality of a system through the number of vulnerabilities of different CVSS severity that are present [72]. This is also a common practice by researchers, e.g., by [78, 116, 147].

## 2.3 Attack graph approaches

An attack graph models how an intruder can traverse a network of nodes through vulnerabilities on the nodes and interconnections between them [110]. Each path in an attack graph concerns a series of exploits, or actions, that leads to a certain undesirable state, e.g., that an intruder gains access to confidential information [153]. Originally, attack graphs were composed manually by red teams; however, as this process is tedious, error-prone and impractical [153] various efforts have been spent to automate it.

According to a recent survey by [101], there are more than 30 types of attack graph approaches published, and a vast number of proposed tools within these categories. This section focuses on describing the more mature approaches that have been published. Notable work that are not detailed include, for instance, the model by Byres and Leversage [106], which adds attack graph capability to the time to compromise model presented in [120], the model by Sheyner and Wing [152], which couples IDSs to attack graphs, and various other efforts [63, 102, 135, 173, 180].

MulVAL [76, 77, 80, 132] uses the output from network vulnerability scanners to model possible attacks on IT architectures. In MulVAL, each vulnerability is associated with a probability that represents how likely an attacker is to successfully exploit it [77]. These probabilities are derived from each vulnerability's denoted access-complexity value according to the Common Vulnerability Scoring System (CVSS) v2 [121] and intuition by the authors. As MulVAL bases its estimations on the output from vulnerability scanners, important attacks (e.g., social engineering and discovery of novel vulnerabilities, i.e., zero-days') and the effectiveness of defenses (e.g., anti-malware and host firewalls) need to be depicted manually by the end-user.

NetSPA [81] and its successors GARNET [184] and NAVIGATOR [35] are similar to MulVAL in the sense that they base their attack graphs on output from network vulnerability scanners. However, they differ in the regard that they treat all identified vulnerabilities as directly exploitable by the attacker. This is an unrealistic assumption as various factors related to successful exploitation are not gathered by network scanners (e.g., effectiveness of intrusion detection systems and attacker knowledge). NetSPA models zero-day attacks by assuming that each depicted software is vulnerable.

TVA-tool [84] (made available commercially as Cauldron [85]) is similar to MulVAL and NetSPA in terms of required data collection and produced results, but uses a database of exploits known to the attacker instead of a database of vulnerabilities. Each exploit is associated with pre- and post-conditions that describe when it can be applied and what state that is reached after exploitation. These exploit conditions are gathered from the commercial database Symantec DeepSight [129].

$k$-Zero Day Safety [181] extends the attack graph reasoning used by MulVAL and TVA-tool with the ability to model zero-day attacks in a similar fashion to what is used by NetSPA. In short, this method computes the $k$ number of zero-days required to compromise an asset by assuming that each modeled service is vulnerable to zero-day attacks. The tool is able to automatically identify means of reducing $k$, e.g., by varying the degree of known vulnerabilities in the network (i.e., patching efficiency). However, as noted by the authors, this model should be extended with the ability to weight different zero-day vulnerabilities as not all are equally exploitable or equally likely to occur.

Frigault et al. [62, 63] propose an attack graph method that is based on Bayesian Networks. The authors utilize TVA-tool for computing "raw" attack graphs and the CVSS for estimating the relative severity of each vulnerability. As the CVSS Base Score ranges from 0-10 the authors propose dividing it by 10 to yield a probability (no empirical evidence is however given to support this translation). The authors also provide a method for incorporating the CVSS Temporal Metrics into this probability. However, as vulnerabilities are not required to be scored in respect to Temporal Metrics, these are rarely available in practice. Conditional probability tables (CPTs) are created using these probabilities in combination with exploit conditions gathered by TVA-tool. If the pre-conditions for an exploit are fulfilled, the probability of using it depends solely on the value associated with it and the number of alternate paths of reaching it. Unfortunately, this means that the method is unable to capture a plethora of common real-world scenarios. For instance, that the effectiveness of an IP-based black list and an intrusion detection system often partially overlap.

ADVISE [59, 105] automatically generates attack graphs representing how an adversary is likely to attack portrayed architectures based on attack steps manually depicted by system experts and adversary experts. As attack steps are manually depicted, very granular aspects can be modeled, for instance, the level of skill an adversary possesses in performing SQL injection attacks. However, it is neither a simple nor quick task to manually compose attack steps relevant to a particular architecture.

The contribution presented in this thesis differs from these tools in the sense that it employs the type of architectural modeling that typically is used in the area of Enterprise Architecture (e.g., ArchiMate [103]). In short, this means that it couples attacks and defenses to objects that the end-user can easily model and understand. It also differs from these tools in respect to its large variety of modeled defenses. It differs from MulVAL, NetSPA, TVA-tool, $k$-Zero Day Safety, and the work by Frigault et al. in the sense that it provides a more holistic, but less granular, set of attacks that are automatically depicted based on the specified object model. It differs from MulVAL, NetSPA, TVA-tool, $k$-Zero Day Safety and ADVISE in the sense that all attack steps and defenses are related probabilistically by Bayesian Networks. It differs from the work by Frigault et al. in the sense that it does not make any overarching assumptions to express CPTs of different phenomena - it instead contains a large set of unique predefined CPTs (one for each attack step and defense).

## 2.4 The Cyber Security Modeling Language

The Cyber Security Modeling Language (CySeMoL) [159, 162], developed at the Royal Institute of Technology (KTH), is a probabilistic relational model (PRM) [61] for estimating the cybersecurity of enterprise-level system architectures, with special focus on Supervisory Control and Data Acquisition (SCADA) systems [167]. A PRM specifies how a Bayesian network [88] should be constructed from an object model, which in turn is depicted according to the constraints of a class model.

CySeMoL includes theory on how attacks and defenses relate quantitatively; thus, security expertise is not required from the user of the PRM. Users must only model their system architecture (e.g., services, operating systems, networks, and users) and some characteristics of these assets (e.g., if an operating system has a host firewall enabled) in order to enable calculations. In total, CySeMoL contains 22 assets, 102 attacks and defenses, and 32 asset relationships. For all attacks in CySeMoL, it is assumed that the attacker is a professional penetration tester who spends one work-week carrying out the attack.

Significant attacks (and corresponding defenses) covered by CySeMoL includes zero-day discovery [163], memory corruption attacks [74, 164], intrusion detection [158], denial of service attacks [161], configurational vulnerabilities [160, 186], attacks on password protection [31, 44, 117] and social-engineering attacks [48, 82, 166].

The contribution in this thesis builds on CySeMoL and adds significant improvements to the model in terms of correctness (papers 1, 8), scope of attacks and defenses (papers 3-7), variable attacker effort (papers 2,8), ease of use and performance (paper 8).

# Chapter 3

# Thesis contribution

This chapter describes the main contribution of this thesis, hereafter referred to as the Predictive, Probabilistic Cyber Security Modeling Language ($P^2$CySeMoL). $P^2$CySeMoL is based on CySeMoL [162], an attack graph model that the author of the present thesis also contributed to. An overview of $P^2$CySeMoL is provided in Section 3.1.

The primary purpose of $P^2$CySeMoL is to support enterprise decision makers with vulnerability estimates and mitigation suggestions. CySeMoL was created for the same purpose; however, it contains limitations that hinder its ability to comprehensively model modern enterprise architectures. The contribution of $P^2$CySeMoL is that it addresses these limitations and improves upon an already versatile tool for vulnerability estimates of enterprise architectures.

A problem with CySeMoL concerns its employed formalism - PRMs. While PRMs certainly have their merits [61], they lack the flexibility necessary for many useful computations. For this purpose, $P^2$CySeMoL has been implemented in the Predictive, Probabilistic Architecture Modeling Framework ($P^2$AMF) [93]. This first contribution is described in Section 3.2.

Due to time-constraints and CySeMoL's focus on SCADA systems, it lacks the ability to model `Assets`, `AttackSteps`, `Defenses` and connections between these that commonly exist in enterprise architectures. For $P^2$CySeMoL, effort has been spent to fulfill these gaps; this second contribution is described in Section 3.3.

A third problem with CySeMoL is that its estimates only are valid for a single work-week spent by a single attacker. $P^2$CySeMoL allows the user to specify an arbitrary amount of time spent by one or more attackers; this third contribution is explained in Section 3.4.

A fourth contribution of this thesis concerns the difficult task of validating of security estimation models such as $P^2$CySeMoL; this is presented in Section 3.5.

Finally, Section 3.6 discusses the practical utility of the contribution.

## 3.1 Overview of contribution

Kordy et al. [101] present a taxonomy consisting of 13 aspects that can be used to categorize attack graph approaches. The overall characteristics of $P^2$CySeMoL according to this taxonomy can be seen in Table 3.1 (see [101] for an extensive description of it). In short, $P^2$CySeMoL is a directed acyclic attack graph model that estimates the likelihood that one or more professional penetration testers, who each have some designated time to spend, are able to succeed with different attack steps against some enterprise architecture. These likelihood estimates are produced based on documented success rates of the correspond-

ing attacks given different circumstances. The user of P$^2$CySeMoL does not need any IT security knowledge; merely knowledge of how the enterprise architecture in question is composed. P$^2$CySeMoL has been tested in real-world case studies and has been implemented in a software tool[1]. P$^2$CySeMoL builds on a previous modeling framework, CySeMoL [162], which first appeared in press during 2010 and is described in Section 2.4. Consequently, a reasonable starting point for anyone who wish to understand the background and foundation of P$^2$CySeMoL can be found by studying CySeMoL. So far, six papers have been written specifically mentioning P$^2$CySeMoL or CySeMoL. This number however excludes papers that have evolved the frameworks without necessarily mentioning them, e.g., master theses and papers that populate the model with measurements of some phenomenon (e.g., papers 3-7 in the present thesis). If counting any paper conducted with P$^2$CySeMoL in mind, the total count is 33.

Table 3.1: P$^2$CySeMoL characteristics according to the taxonomy by [101].

| **Aspect** | **P$^2$CySeMoL** |
|---|---|
| Attack or defense | Integrates attack and defense modeling |
| Static or sequential | Supports time and order dependencies |
| Quantification | Supports numerous generic and diverse metrics |
| Main purpose | Support risk assessment |
| Extensions | New connectors, extended graph structure |
| Structure | Directed acyclic graph |
| Connectors | Derived and depicted connections according to the Object Constraint Language (OCL) |
| Formalization | Bayesian networks and OCL queries |
| Tool availability | A prototype tool exists |
| Case study | Real and fictional case studies have been documented |
| External use | People and institutions who did not invent the formalism have used it |
| Paper count | 6 (33) |
| Year | 2010 |

## 3.2 Implementation in a new framework

### Predictive, Probabilistic Architecture Modeling Framework

To enable attack graph modeling and calculation, there is need of a framework that dictates how attacks and defenses relate. For this purpose, P$^2$CySeMoL employs P$^2$AMF [93]. In short, P$^2$AMF combines the Object Constraint Language (OCL) [174] with probabilistic analysis to enable assessment and prediction of system properties. The main feature of P$^2$AMF is its ability to express uncertainties of objects, relations and attributes in Unified Modeling Language (UML) models and perform probabilistic assessments incorporating these uncertainties.

The probabilistic aspects are considered in a Monte-Carlo fashion: First, the user specifies a desired number of samples. Thereafter, a set of object models corresponding to the chosen sample size is created. The stochastic variables of the class model are instantiated

[1] www.ics.kth.se/eaat

with instance values according to their respective designated distribution. This includes the existence of classes and relationships, which are instantiated on a frequency depicted by the corresponding probability distributions. Then, each P$^2$AMF statement is transformed into an OCL statement that can be evaluated by the OCL parser. Once the evaluation of all samples has been performed, results are aggregated and visualized according to the design of the class model.

### Framework of contribution

An overview of the P$^2$CySeMoL implementation in P$^2$AMF can be seen in Figure 3.1. There are four types of classes in P$^2$CySeMoL: `Attacker`, `AttackStep`, `Defense` and `Asset`. Each `AttackStep` and `Defense` is connected to an `Asset` that it compromises or protects. These connections are not required to be specified by a user of P$^2$AMF; the user is only required to depict and connect `Assets`, and if needed, specify the state of different `Defenses`. An example of this can be seen for the classes related to the P$^2$CySeMoL concept `NetworkZone` (a connection point of various IT devices, e.g., a Local Area Network [LAN]) in Figure 3.2. When the `Asset NetworkZone` is depicted in an object model, the three `AttackSteps` and two `Defenses` connected to it are depicted as well.

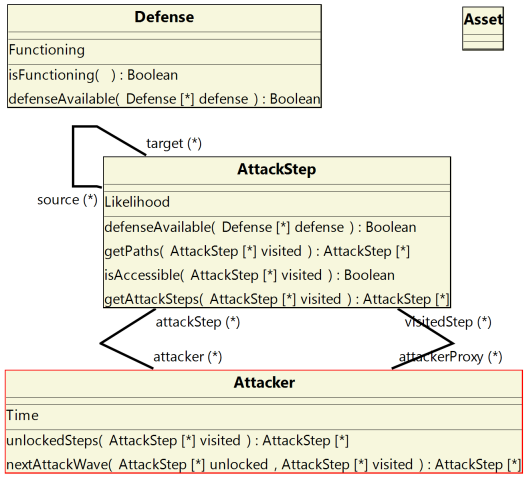


Figure 3.1: `Attacker`, `AttackStep`, `Defense` and `Asset`. OCL operations are given in the lower box of each class; attributes are given in the upper box of each class.

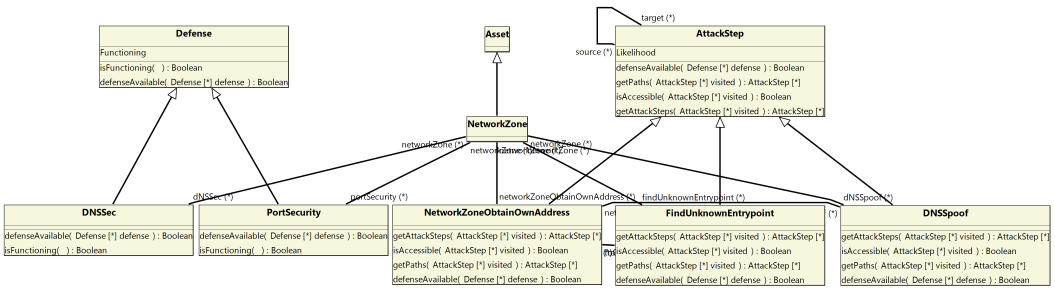


Figure 3.2: Classes corresponding to `NetworkZone`.

Connections between `AttackSteps` are derived automatically depending on how the user has connected `Assets` in an object model. For example, if a user has connected two `NetworkZones` *NZ1* and *NZ2* to a `NetworkInterface` *NI* (e.g., a router), then there will be a derived connection from the `AttackStep` `NZ1.ObtainAddress` to the `AttackStep` `NZ2.FindUnknownEntry` (see Figure 3.3).

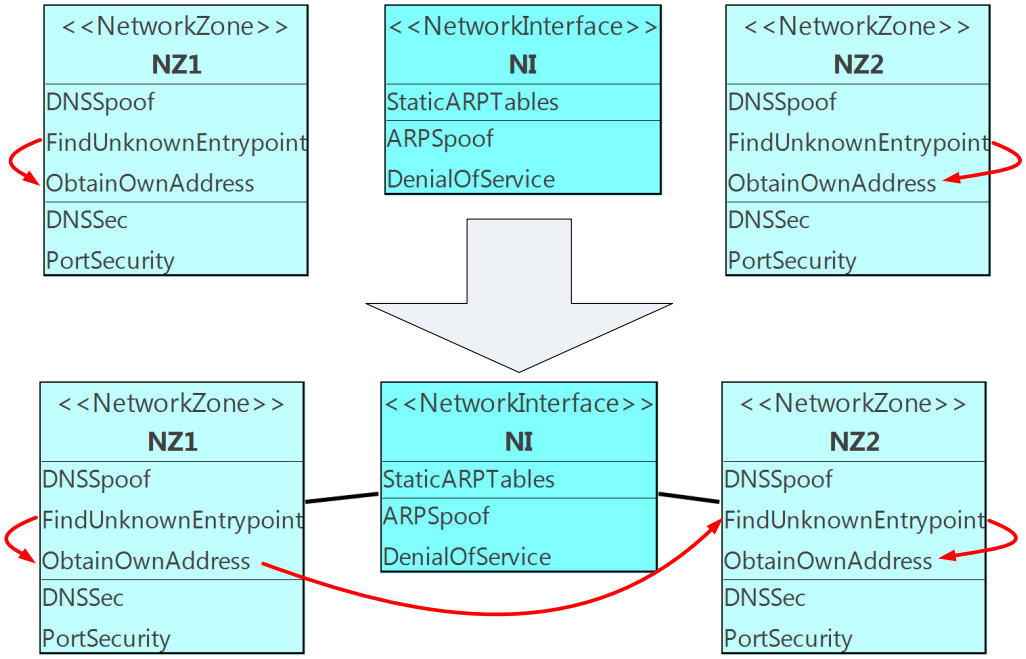


Figure 3.3: Example describing derived connections between `AttackSteps` (these are denoted by red arrows).

The OCL operations serve to compute the likelihood of one or more `Attackers` being able to reach the different parts of an object model through any means possible (using the derived connections between `AttackSteps` and probabilistic logic present for each `AttackStep`). Any `AttackStep` that an `Attacker` is connected to is considered source for the attack; it evaluates to TRUE regardless of the properties of the object model in question.

`AttackStep.Likelihood` denotes the probability of an attacker being able to successfully utilize a particular `AttackStep`; it is the fraction of samples where that particular `AttackStep` could be reached. `Defense.Functioning` denotes the likelihood that a `Defense` is available. `Attacker.Time` denotes how much time (in work days) that the attacker has available for an attack.

The P$^2$AMF implementation of P$^2$CySeMoL enables a range of improvements compared to CySeMoL, in particular:

- Its performance scales linearly with the number of `AttackSteps` in an object model.
- It does not require specifying the target of an attack as it considers *all* `AttackSteps` as targets. This allows computing and presenting the entire vulnerability of an object model using a single calculation.
- It can model an arbitrary amount of attackers, with arbitrary number of connected `AttackSteps`.

The framework and its contributions are described in detail in paper 8.

## 3.3 Scope of contribution

An overview of the scope of the P$^2$CySeMoL class model can be seen in Figure 3.4. Here, `AttackSteps` and `Defenses` are coupled to their corresponding `Assets`. Derived connections between `AttackSteps` and `Defenses` are for presentation purposes not illustrated in Figure 3.4. `Attacker` (with the attribute `Time`) is due to the same reason not shown (it can be related to any `AttackStep`).

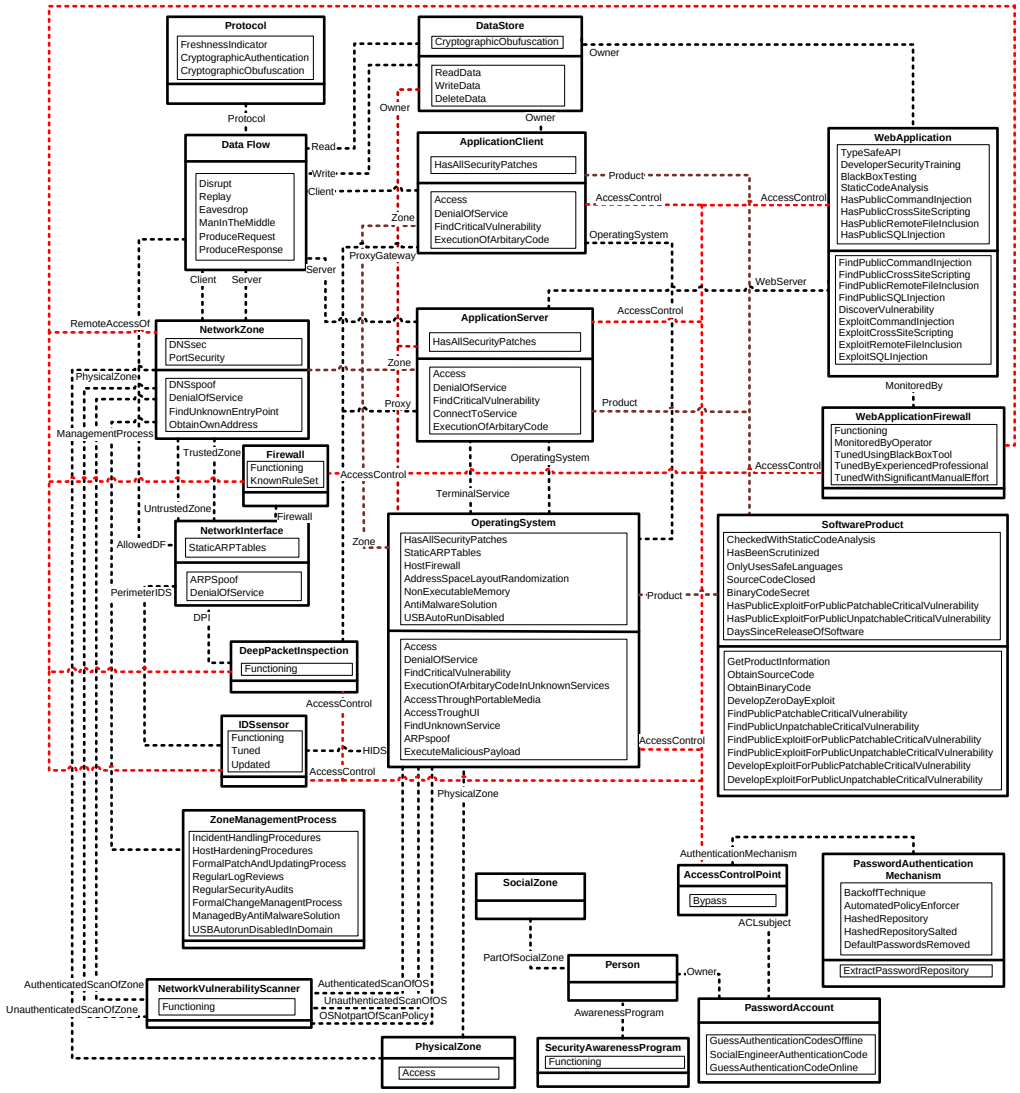


Figure 3.4: An overview of the P$^2$CySeMoL class model. The upper box of an asset contains the defenses associated with it. The lower box contains the attack steps associated with the asset. Colors are used only to make asset relations more clear.

As P$^2$CySeMoL is founded upon CySeMoL, it inherits much of its scope:

- Its 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.
- It models objects of relevance for enterprise decision making.
- It does not require that its users are security experts.

P$^2$CySeMoL also inherits the majority of CySeMoL's class model. However, partly due to CySeMoL's focus on SCADA systems, and partly due to time constraints during its creation, CySeMoL is not sufficient for modeling enterprise architectures. P$^2$CySeMoL extends the scope of CySeMoL in a various ways; an overview of the most significant extensions in terms of added `Assets`, `AttackSteps` and `Defenses` are shown in Table 3.2 and described in the remainder of this section.

Table 3.2: Overview of added `Assets`, `AttackSteps` and `Defenses`

| **Asset** | **Added attack steps and defenses** |
|---|---|
| `WebApplication` | Completely new concept |
| `WebApplicationFirewall` | Completely new concept |
| `SocialZone` | Completely new concept |
| `NetworkVulnerabilityScanner` | Completely new concept |
| `IDSSensor` | Probabilities for detection of arbitrary code execution attacks have been updated |
| `Firewall` | `KnownRuleSet` |
| `ZoneManagementProcess` | `ManagedByAntiMalwareSolution` and `USBAutoRunDisabledInDomain` |
| `OperatingSystem` | `ExecuteMaliciousPayload`, `AntiMalwareSolution` and `USBAutoRunDisabled` |
| `SoftwareProduct` | `HasPublicExploitForPublic-PatchableCriticalVulnerability`, `HasPublicExploitForPublic-UnpatchableCriticalVulnerability`, `DaysSinceReleaseOfSoftware`, `FindPublicExploitForPublic-PatchableCriticalVulnerability`, `FindPublicExploitForPublic-UnpatchableCriticalVulnerability`, `DevelopExploitForPublic-PatchableCriticalVulnerability`, `DevelopExploitForPublic-UnpatchableCriticalVulnerability`, `FindPublicPatchableCriticalVulnerability` and `FindPublicUnpatchable-CriticalVulnerability` |
| `AccessControlPoint` | Can now also be related to `IDSsensor`, `DeepPacketInspection`, `Firewall` |

SCADA systems have many similarities to traditional business IT [167]. However, there are exceptions that have resulted in delimitations regarding the scope of CySeMoL. In particular, integrity and availability are generally viewed as more important than confidentiality as unavailability (i.e., failure to observe and control the state of the physical process) can have severe effects for the society at large [14]. As a result, various common cyber security tools that can have a negative impact on availability are not used in SCADA systems, or part of CySeMoL. For instance, anti-malware and network scanners can impact the availability of IT components, and thus need to be used with care [167, 51], or sometimes not at all [126] in SCADA architectures. In P$^2$CySeMoL, these tools are possible to model by the user, satisfying SCADA and regular business IT alike.

Another major addition is the concept of `SocialZone`; a group of individuals who are prone to sharing documents and devices, e.g., a work-group in an office space. `SocialZone` enables modeling attacks against IT-wise isolated devices (which often is the case in information-critical environments, e.g., critical infrastructure control systems). In practice, this is managed by connecting `Access` of an `OperatingSystem` to `AccessThroughPortableMedia` of other `OperatingSystems` that have local users who share the same `SocialZone`.

To enable quantitative analysis incorporating the extensions shown in Table 3.2, various sources of quantitative data have been consulted. Some data were readily available through previous studies by other researchers. For instance, regarding the effectiveness of anti-malware solutions (e.g., [18, 125]) and regarding code injection using USB drives (e.g., [82]). When no data was available, new empirical studies were conducted to extend CySeMoL's scope; these are described in the next four subsections and elaborated in papers 3-7.

It is worth mentioning that this research not only involved extending CySeMoL; it also involved removing `AttackSteps`. In particular, CySeMoL differentiates between "*OtherZone*" and "*SameZone*" (attacks conducted within a network zone compared to between network zones) in order to increase computational performance (only a single attack type can be conducted within the same network). This concept is not necessary to keep in P$^2$CySeMoL as P$^2$CySeMoL's performance scales linearly with the number of `AttackSteps` in an object model. With this concept removed, results are not only more realistic, but also less confusing.

### Attacks against web applications

While considerable effort has been spent by both academia and industry to mitigate web application vulnerabilities, a recent study [144] shows that vulnerabilities still are frequent. This study also shows that most vulnerabilities can be prevented by straight-forward validation mechanisms based on common data types.

A reason behind this could be the lack of useful data on the effectiveness of different web application security measures. The empirical studies that do exist (e.g., [15, 50, 55, 57, 168]) focus on the effectiveness of technical defenses in isolation (black-box vulnerability scanners in particular), when combinations of defenses frequently are used in practice. Furthermore, these studies do not consider common measures considered by enterprise decision makers, e.g., developer security training. Papers 3 and 4 describe empirical studies used to extend P$^2$CySeMoL in terms of security measures used during both the development (paper 3) and the operation (paper 4) of web applications.

Paper 3 extends P$^2$CySeMoL with the `Asset WebApplication`, which needs to be connected to an `ApplicationServer` that runs it (i.e., a web server). This enables the possibility of discovering a `CommandInjection`, `CrossSiteScripting`, `RemoteFileInclusion`, or `SQLInjection` vulnerability in a `WebApplication` given the presence or

absence of four `Defenses`: `TypeSafeAPI`, `DeveloperSecurityTraining`, `BlackBoxTesting` and `StaticCodeAnalysis`.

If an attacker is able to find a vulnerability, (s)he can attempt to exploit it. An SQL injection vulnerability can be exploited to read, write or delete data in a `DataStore`. Command injection, remote file inclusion and SQL injection vulnerabilities can be exploited to `ExecuteMaliciousPayload` in an `OperatingSystem` operating the `ApplicationServer` that runs the `WebApplication`. A cross site scripting vulnerability can be exploited for `ExecutionOfArbitaryCode` in an `ApplicationClient` (a web browser) exploring the content of the compromised `WebApplication`.

Paper 4 extends P$^2$CySeMoL with the `Asset WebApplicationFirewall`. If a `WebApplicationFirewall` is connected to a `WebApplication`, it has a possibility of preventing attacks against it. The likelihood of prevention depends on whether the `WebApplicationFirewall` is `MonitoredByOperator`, `TunedUsingBlackBoxTool`, `TunedByExperiencedProfessional` or `TunedWithSignificantManualEffort`.

### Acquisition of critical vulnerabilities and exploits

CySeMoL does not distinguish between software vulnerabilities and exploits. Furthermore, it does not consider that an attacker might wait for public vulnerabilities and exploits to appear in the public domain. Paper 5 mitigates these issues; this paper describes a study where 13 states on the topic of vulnerability and exploit acquisition are related quantitatively by 17 activities. The corresponding `AttackSteps` and `Defenses` are incorporated in the `Asset SoftwareProduct`.

### Automated network vulnerability scanning

A network vulnerability scanner is a commonly used tool to identify vulnerabilities such as unpatched software and weak passwords. CySeMoL does not model these as they sometimes cause unavailability problems, and thus rarely are used in SCADA environments [51]. However, they are frequently used in office environments, and thus essential to model to fulfill the purpose of P$^2$CySeMoL.

Paper 6 extends P$^2$CySeMoL with the `Asset NetworkVulnerabilityScanner` and data on the general effectiveness of the tool. This `Asset` can be connected to a `NetworkZone` or `OperatingSystem` denoting either an authenticated or unauthenticated scan. During an unauthenticated scan, the scanner probes for vulnerabilities that are testable through TCP or UDP without any privileges on the studied systems - i.e., any `ApplicationServer` (e.g., an FTP service) connected to the probed `OperatingSystem`. If an `ApplicationServer` has a login interface (e.g., SSH or FTP), the scanner can also attempt to evaluate any poor passwords for this interface. During an authenticated scan, the scanner is allowed to log in to the probed systems. Thus, an authenticated scan is typically both more effective and can not only evaluate vulnerabilities for the `ApplicationServer`, but also for any `ApplicationClient` (e.g., a web browser) residing on the probed `OperatingSystem`. Both scanning types can also help find `ApplicationServers` unknown to the network administrator (`OperatingSystem.FindUnknownService`). An `OperatingSystem` can also be designated to *not* be part of the scanning policy (this is common in practice as scans can cause availability issues).

### Signature-based network intrusion detection systems

CySeMoL contains estimates on the general effectiveness of signature-based network intrusion detection systems (SNIDS) at detecting arbitrary code execution attacks [158]; however, these are based solely on expert judgment. Paper 7 presents an experiment employed to refine these data. This study is an example of how quantitative data in P$^2$CySeMoL can be updated without requiring any change to connections between `AttackSteps` and `Defenses`. Both a network SNIDS's ability to detect zero-days' and standard attacks given a `Tuned` SNIDS, which rule set either is or is not `Updated` are renewed by paper 7.

## 3.4 Modeling variable attacker effort

All quantitative estimates in CySeMoL are made based on the assumption that the attacker has a single work week to spend. This is problematic as 1) it is unknown how much attacker effort that is reasonable in general and 2) it is certain that some enterprises are more attractive to attackers than others (e.g., a bank compared to a personal web page). Thus, it would be valuable to calculate the probability of attacks given different amounts of effort spent by an attacker. P$^2$CySeMoL does this by sampling based on probabilities elicited from cumulative density functions (CDFs). As the outcome of these operations are deterministic, P$^2$AMF only executes them once, rather than for each sample. An example of how time-based sampling is implemented in P$^2$CySeMoL can be seen in Algorithm 1. Here, `bernoulli(exp(0.0715,Attacker.Time))` is equivalent to $P = P(X \leq \texttt{Attacker.Time} | X \in \text{Exponential}(0.0715))$. For example, given `Attacker.Time` = 5 days, $P = 30\%$. This is further described in paper 8.

```
let awarenessprogramTrue : Real = bernoulli(exp(0.0715,Attacker.Time))
let awarenessprogramFalse : Real = bernoulli(exp(0.241,Attacker.Time))
if visited− >intersection(self.interface)− >notEmpty() then
    if self.person.awarenessprogram.functioning then
        awarenessprogramTrue
    else
        awarenessprogramFalse
    end
else
    False
end
```

**Algorithm 1:** `SocialEngineerCredentials.isAccessible`

For this procedure to work, there is a need to compose CDFs that well describe the data of interest. Unfortunately, for some datasets (e.g., [164]), distribution fitting was not feasible. Given this scenario, there is a need to assume some CDF, preferably based on empirical results from similar studies. Paper 2 provides support regarding this property by presenting what distribution that is best fit for modeling the time required to compromise a computer system in general. More specifically, it studies the time from installation of a computer system to compromise of that system, using a dataset of all malware incidents across 260,000 computers over three years. The results from the study show that the two-parameter generalized Pareto (PAR) is best suited for this purpose. Paper 2 further examines the validity of this finding by examining which distribution that is best fit for modeling the dataset described in paper 1; the findings from this study also suggest that

PAR is the best fit. Consequently, when applicable, PAR was chosen to model uncertain variables.

## 3.5 A method and dataset for validation

To validate cybersecurity models is a difficult task as both data and methods available for this purpose are scarce. Paper 1 provides a method and dataset that can be used to validate cybersecurity estimation models. This method is essentially the same as the concept of Net Working Time (NWT) [107]; the time spent attempting to break into a safe by testers using specified sets of tools, such as diamond-grinding tools and high-speed carbide-tip drills. Different safes' NWT can be compared to determine the relative security level of each safe. In the context of paper 1: to compare the time required by one or more attackers to compromise some asset with security estimates of this asset given by a model or metric. An asset that is deemed more secure by a model or metric should also require a higher Time to Compromise (TTC). The dataset presented in paper 1 consists of 34 TTC estimates extracted from an international cyber defense exercise with more than 100 participants. In paper 1, this dataset is used to examine the correlation between TTC and security estimates by 18 security metrics, including both academic contributions and commonly practiced methods. The results from this study show that none of the metrics significantly correlate with TTC. In paper 8, this method and dataset are reused to examine the validity of P$^2$CySeMoL.

## 3.6 Practical utility

Atzeni and Lioy [16] present five properties that any security measurement system should exhibit in order to be effective and useful:

- *Succinctness*: Only important parameters should be considered, letting aside aspects not important to the definition and/or the comprehension of the entity under measurement. Such property aims to reduce both a measure's complexity and uncertainty.
- *Repeatability*: If repeated in the same context, with exactly the same conditions, the measure should return the same result.
- *Objectiveness*: The measure should not be influenced by the measurers' will, beliefs, or actual feelings.
- *Easiness*: The measure of an attribute should raise knowledge about the entity itself, sometimes with the purpose of improving the usefulness of the entity. However, if the measure is too difficult to be performed, or simply impossible to accomplish, the knowledge's gain is not sufficient to motivate the measurement.
- *Clarity*: A measure should be easy to interpret, at least in its operative context.

P$^2$CySeMoL has been tailored to only include theory that is of critical importance towards practical vulnerability estimates (see Section 4.2); thus, we argue that the contribution satisfies the criterion of *Succinctness*. P$^2$CySeMoL produces the same result if it is provided the same input, regardless of the measurers will, beliefs, or feelings; thus, it satisfies the criteria of *Repeatability* and *Objectiveness*.

P$^2$CySeMoL has been implemented in a software tool called the Enterprise Architecture Analysis Tool (EAAT) [28]. To perform modeling and analysis with P$^2$CySeMoL in

EAAT, there is a need to: 1) create an object model, 2) connect one or more `Attackers` to one or more `Assets`, and 3) press "calculate". The first step can be managed by dragging and dropping `Assets` and then connecting these in some manner valid to the constraints of P$^2$CySeMoL's class model. It could also be partially automated by feeding a resulting XML from a network vulnerability scan by Nexpose to P$^2$CySeMoL [27, 73]. `Defenses` all have default states corresponding to what is believed to be the most common in practice (e.g., ASLR is TRUE per default as modern operating systems support it) or derived from the existence of other `Defenses` (e.g., `ZoneManagementProcess.FormalPatchAndUpdatingProcess` affects the likelihood of software being fully patched). The number of work days that an `Attacker` is able to spend on an attack is fully customizable by the user (per default, this value is five). Finally, each `Asset` in P$^2$CySeMoL has been color coded to enable more user-friendly overviews (see Figure 3.5 for some examples). With this work, we argue that P$^2$CySeMoL satisfies the criterion of *Easiness*.

When a calculation is complete, the probability of each `AttackStep` in the object model being TRUE is illustrated with a probability distribution and a color on a scale from [0: green, 50: yellow, 100: red]. This probability denotes the likelihood that one of the depicted attackers are able to conduct this `AttackStep`. To allow easy-to-overview results, `Assets` are color-coded based on a profile chosen by the user (this overwrites the default color code of any `Asset` previous to the calculation). Current profiles include, for example, the mean of all corresponding `AttackSteps`, only denial-of-service type `AttackSteps`, only remote access of `Assets` as root/administrator, and no coloring. The visualized profile can be swapped on the fly by the user. An example object model before and after calculation can be seen in Figure 3.5 and Figure 3.6; Figure 3.7 and Figure 3.8 depicts the same object model before and after calculation with `AttackSteps` and `Defenses` shown. This small object model also provides a hint of the wide range of attacks that P$^2$CySeMoL can simulate. For instance, that an attacker on the Internet can find an unknown entry point through the firewall, that the office system can be compromised through code injection of its software or social engineering of the individual using the system, and that the IT-wise isolated system can be compromised due to that the individual using this system shares the same social zone as the individual using the office system (given three days spent on the attack, the likelihood of the attacker gaining access to the isolated system is 23%). What `Assets`, `AttackSteps` and `Defenses` that should be visible in a particular viewpoint can be fully customized by a user of P$^2$CySeMoL. Due to these features, we argue that the contribution satisfies the criterion of *Clarity*.

Apart from main practical utility of the contribution, to estimate vulnerability and mitigation options, there are a number of positive side-effects from usage of P$^2$CySeMoL that can be theorized based on our experiences from case studies of CySeMoL and P$^2$CySeMoL.

Next to every enterprise log their overall IT architecture through some means, often using Microsoft Visio or similar tools. Object models created by usage of these tools suffer from their lack of any viable security ontology, which can result in a number of different issues [124]; for instance, different individuals might depict the same aspect using different notation, or different aspects using the same notation, causing confusion for others' interpreting this model.

Even if the numbers produced by P$^2$CySeMoL are not trusted, it might still prove effective in terms of decreasing vulnerability as it provides an ontology describing what to model, how to model, and how to interpret a model. Furthermore, as many enterprises log their architectures already anyway, it should be a small endeavor to instead do it using P$^2$CySeMoL. Along the same line of argument, P$^2$CySeMoL could be used an an interface to facilitate communication between different stakeholders, and thus manage the common organizational problem of miscommunication [40].

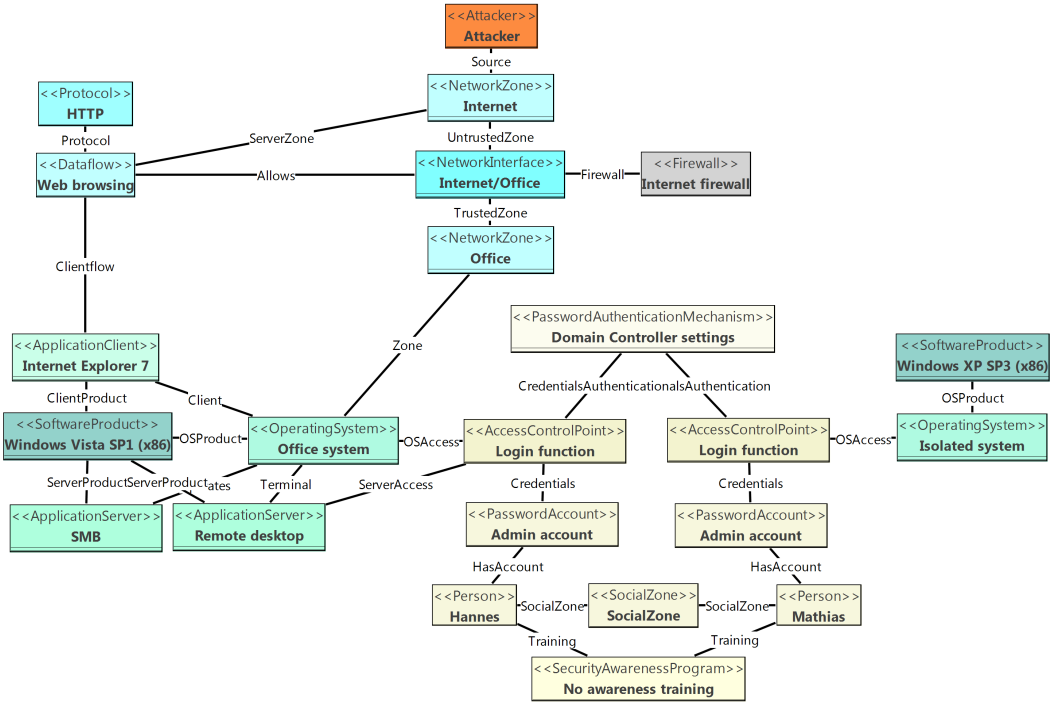


Figure 3.5: An example P$^2$CySeMoL object model.

Finally, P$^2$CySeMoL could be used as a tool for education. Using software tools for security awareness training is something that several previous research efforts have studied and shown useful. For instance, Sheng et al. [150] developed an online computer game aimed to teach users about how to avoid phishing attempts and observed that the amount of correctly identified phishing attempts increased from 69% to 83% after that participants had played the game. A similar educational tool is presented by [37]; however, this tool focuses on end-user security threats in general. To our best knowledge, there is however no educational tool regarding enterprise security threats - which happens to be the scope of P$^2$CySeMoL.

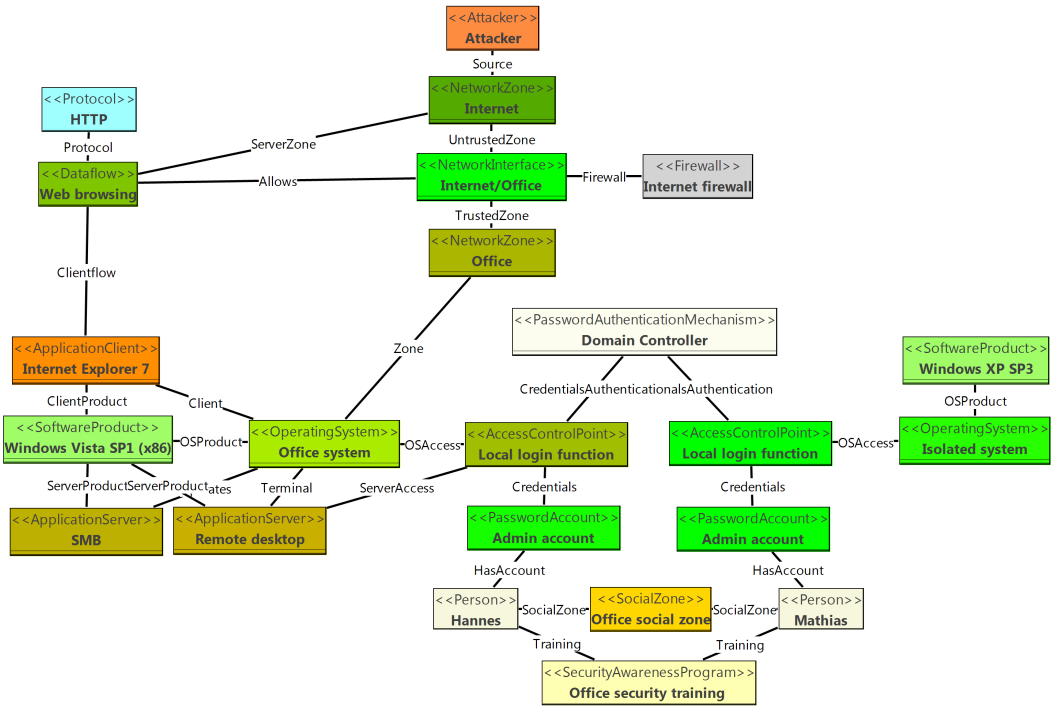


Figure 3.6: Example object model with results shown (given three days spent on the attack and asset coloring based on means).

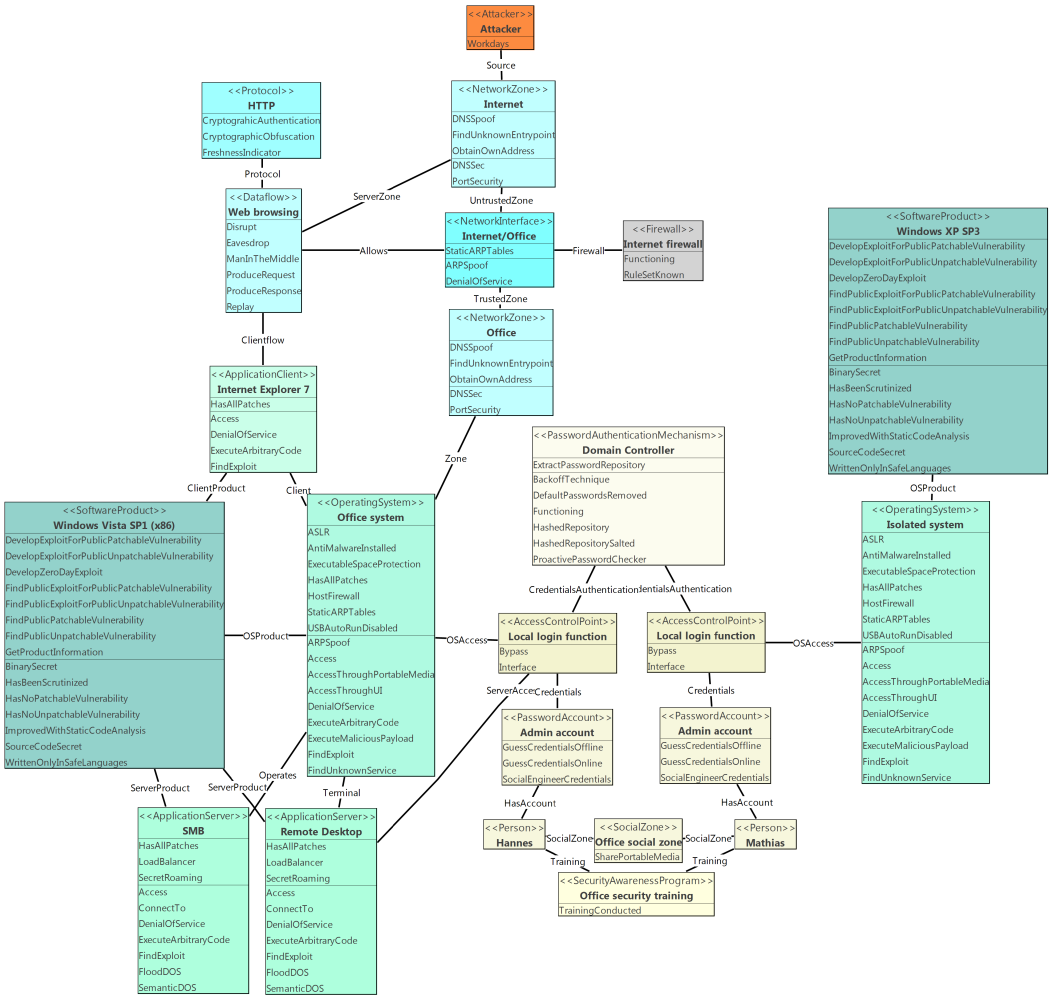


Figure 3.7: Example object model with `AttackSteps` and `Defenses` shown.

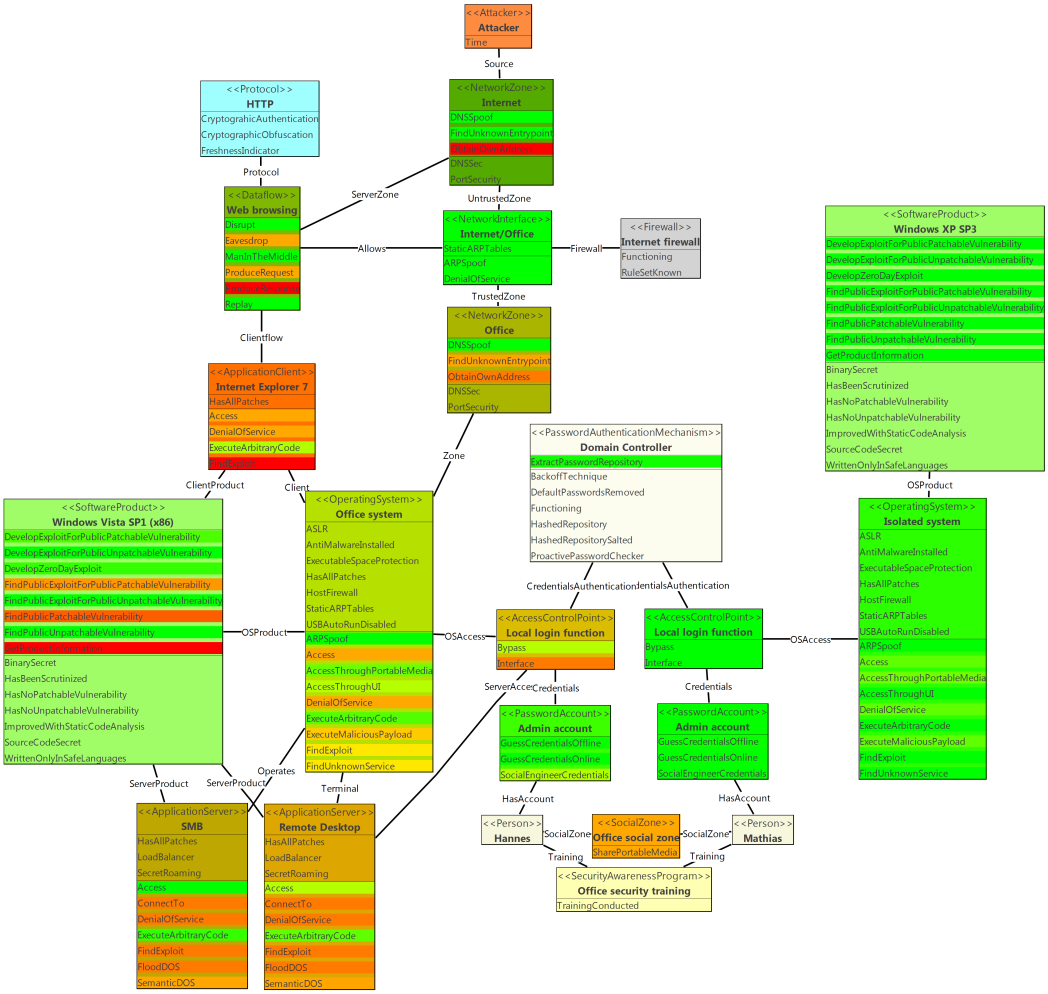


Figure 3.8: Example object model with `AttackSteps`, `Defenses` and results shown.

# Chapter 4

# Research design

The work on this thesis began during 2010, when CySeMoL was first published [159]. CySeMoL was initiated due to the lack of methods for examining the cybersecurity of enterprise architectures; this first publication included a method (PRMs) and a set of abstract classes based on the Common Criteria framework (e.g., `ThreatAgent`, `Asset`, and `AttackStep`).

The next step of CySeMoL's creation involved populating its qualitative structure with `AttackSteps`, `Defenses` and `Assets`, and identifying relations between these. The first step in this process involved creating a preliminary qualitative structure through an extensive literature review. The second step involved performing a large number of interviews with system owners and cybersecurity experts to extract a structure with classes and relations that are valid to the real world. During this step, some classes were removed, and others added.

When a qualitative structure had been composed, the process of populating this structure with quantitative data began. Some data could be derived through logical constraints (e.g., it is impossible to inject code if the attacker cannot reach the vulnerable software). Non-deterministic variables were specified using secondary data on observational studies when these were available (e.g., regarding password cracking [31, 44, 117]), and primary data collection through surveys with domain experts when necessary. The work on CySeMoL culminated during 2012 with a paper [162] and a thesis/dissertation [156].

CySeMoL is to our knowledge the most comprehensive attack graph model available in terms of quantitative enterprise architecture security estimations. However, it does not come without faults. In particular, it lacks the ability to model `AttackSteps`, `Defenses` and `Assets` that are common in modern enteprise architecturs - especially those related to web applications. Furthermore, during CySeMoL's development and testing (e.g., its case studies, research discussions, and paper reviews) a number of important improvement possibilities were discovered. For instance, that it is demanding to compare attack graph results created by CySeMoL, and that its estimates should be valid for any amount of attacker effort; not only a scenario where an attacker spends a single work-week on the attack.

A reasonable means of developing a model for cybersecurity analysis of enterprise architectures is thus to reuse the extensive work that has been put into CySeMoL, and improve upon its flaws. This was also the method for producing the contribution described in this thesis; an overall illustration of how it was managed can be seen in Figure 4.1. Here, the method for creating the $P^2$CySeMoL framework is described in Section 4.1, the methods for creating quantitative security theory in Section 4.2, the methods for enabling variable attacker effort in Section 4.3, and the validation in Section 4.4.

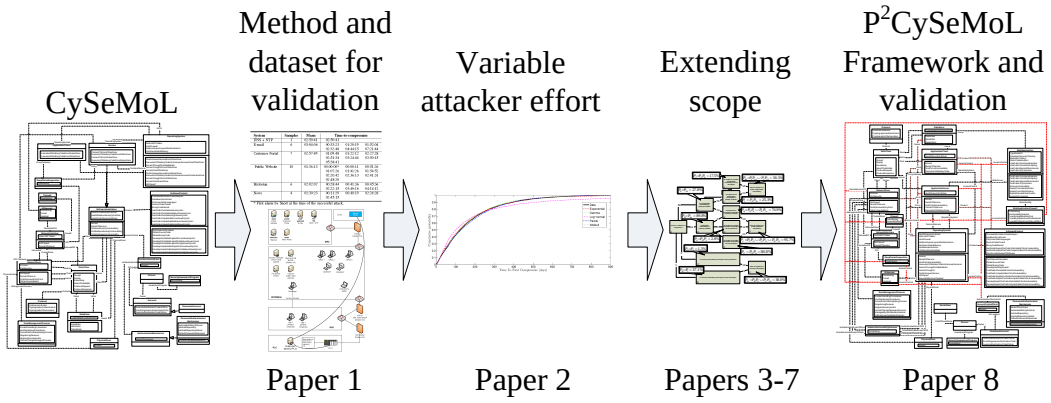


Figure 4.1: Overall research process.

## 4.1 Creation of framework

The overall choices in terms of classes and terminology in P$^2$CySeMoL are based on CySeMoL, and hence also the Common Criteria (along with numerous other literature). Its implementation in P$^2$AMF is a process that began during 2012. Choices made during this process were based on literature reviews and extensive usage through fictional and real-world case studies. Usability features such as color profiles for calculations are also products of this process.

## 4.2 Creation of quantitative security theory

The choice of research strategy in information systems research is not a simple task [177]. It is no different in the IT security domain, where a wide variety of methods have been used to obtain data on different phenomena (see Section 2.1).

Easterbrook et al. [53] presents an overview of the different strategies that can be utilized in empirical software engineering research. Each paper described in this thesis involves some sort of data collection; the strategies employed by them according to the categories in [53] can be seen in Table 4.1. As can be seen, case studies have been used extensively for the present research. The reason behind this methodological choice is that qualitative data from case studies can provide valuable insight in terms of how to carry out and analyze results by quantitative studies [64, 89]. For paper 2, an exploratory case study was employed to gain insight into how incidents were collected and archived, and to understand contextual reasons behind malware trends; for papers 3-7, exploratory case studies were conducted to derive variables, states and tools useful to the purpose of the overall research (to support enterprise decision making); for paper 8, two confirmatory case studies were carried out to examine the usefulness of P$^2$CySeMoL as a whole. For instance, three interviews and a workshop with three respondents were conducted to elicit variables and states for the surveys presented in papers 3 and 4, on an abstraction level useful for enterprise decision making [71].

Of the quantitative methods employed to extend the security theory of P$^2$CySeMoL, papers 1, 6 and 7 describe controlled experiments, and papers 2-5 present survey research. More detailed descriptions of these follows next.

Paper 1 features data gathered from an international cyber defense exercise with more

Table 4.1: Chosen research strategies, following [53].

| | Paper | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Strategy** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Controlled experiments | x | | | | | x | x | |
| Case studies | | x | x | x | x | x | x | x |
| Survey research | | x | x | x | x | | | |
| Ethnographies[a] | | | | | | | | |
| Action Research[a] | | | | | | | | |

[a]Not employed

than 100 participants. Data was collected using TCP-dump sniffers and parsed by the SNIDS Snort to identify malicious traffic. As attackers could be identified through their corresponding MAC addresses, activity by defenders could be filtered out. TTC was measured as the calendar time from first Snort alarm for a system until successful compromise of it; the latter was discovered based on logs from attackers, defenders and observers. Of the systems employed in the exercise architecture, six were targeted during the entire exercise and had static environments in terms of what services that were required to be operated by the defenders. Attacks against these were chosen to be studied, providing a total of 34 TTC values for analysis.

Paper 2 presents archival data on 5.6 million malware events gathered from approximately 260,000 computer systems during a period of three years. Time to first compromise was measured as the time of first installation of a system in the enterprise until the time of the first malware alarm observed on this system.

Paper 3 describes a study where the judgment of 21 experts was utilized to estimate the effort required by a professional penetration tester to discover a critical vulnerability in a web application that had been developed with or without the aid of four common security mechanisms. Experts were invited to a web survey based on participation in relevant email lists. The possibility of non-experts participating was handled by Cooke's classical method [38] as this method weights the answers of respondents based on their performance on a (unknown to the respondents) knowledge test. Expert elicitation is a common means of creating Bayesian models when available data is sparse compared to the number of nodes that need be populated [92].

Paper 4 features the same approach as paper 3, but instead estimates the effectiveness of web application firewalls, a type defense commonly used to improve the security of web applications in operation. In this study, experts were invited not only based on participation in relevant email lists (as paper 3), but also based on whether they had authored any peer-reviewed publications on the topic. A total of 49 experts fully answered the web survey and were scored by Cooke's classical method.

Paper 5 gathered data relevant to vulnerability and exploit acquisition from previous empirical studies, vulnerability databases and a survey with 58 experts. This expert elicitation process used a similar web survey as paper 3 and 4, but using a different sampling frame. Here, only those who had actually been credited for the discovery of vulnerabilities were invited. Consequently, usage of Cooke's classical method was judged unnecessary for this study.

Paper 6 describes an experiment where seven of the more commonly employed commercial scanners were studied in terms of, in particular, missed vulnerabilities and mitigation suggestions (false negatives), and how many non-existent vulnerabilities that a scanner in-

correctly claims exist (false positives). A total of 50 false negatives and 40 false positives per scanner were analyzed, all being vulnerabilities of CVSS High severity (i.e., a CVSS score higher than 7). The experimental architecture was designed by a group of computer security specialists and computer security researchers, incorporating a total of 28 different systems running a diverse set of applications.

Paper 7 presents an experiment that involved 356 attacks and the industry-standard SNIDS Snort. Of the tested 356 attacks, 183 were more novel than the used Snort rule set and 173 older than the rule set. The results from this study thus both hint towards SNIDS effectiveness at detecting both attacks more novel than the rule set (i.e., zero-days') and "standard" attacks. The study then analyzed how the zero-days' were detected, how prone the corresponding rules are to false alarms, and how easily these rules can be circumvented.

Paper 8 does not include any new quantitative data. However, it incorporates the essence of the data presented in papers 1-7.

## 4.3 Sampling based on attacker effort

A number of methods were used to enable modeling variable attacker effort (measured as time) spent. First, some quantitative data in CySeMoL were readily available as time-based. For instance, the effort required to discover a zero-day vulnerability was measured on a scale of work days [163]. For these datasets, Akaike Information Criterion [2] was used to find the distribution best fit for modeling its properties, with tested distributions selected based on choices and findings from paper 2. If no distribution was well fit, some kind of interpolation method was employed to derive a custom CDF; typically, linear interpolation as the questions of the expert surveys used to populate CySeMoL and P$^2$CySeMoL were presented to respondents with graphs depicting linearly connected dots (see Figure 1 in paper 3 for an example).

Some datasets have very limited variety of different time estimations, prohibiting both distribution fitting and any reasonable interpolation methodology. For instance, the likelihood of successful arbitrary code execution was estimated using a constant time spent preparing for the attack (one work week) [164]. For these datasets, paper 2 provided empirical justification towards what distribution to assume. If possible, the two-parameter generalized Pareto (PAR) was chosen as this was found best fit. When only a single datapoint of a CDF is available (e.g., [164]), there are however an infinite number of parameter solutions for PAR. For such data, the one-parameter generalized Pareto, also known as the exponential distribution, was employed instead (that there is only a single solution for). This choice is supported by [95], who found the exponential distribution well fit for modeling TTC.

## 4.4 Validation

Showing that a model or metric is "correct" is essential towards its utility as an incorrect system can cause costly errors [130]. In theory, a security metric can *prove* the absence of vulnerabilities [178]. In practice, however, the great complexity of the issue means that no methodology can provide absolute assurance that a system is secure [21, 178]. A model's correctness might although be sufficient as long as its estimates, while being inaccurate, point to the right conclusions [130]. In other words, the exact values provided by P$^2$CySeMoL might not be terribly important. Rather, it is important that the relative ranking in terms of vulnerability is reasonable.

O'Keefe et al. [130] argue that an expert system can be validated both on a component level and on a system level. Component level validation concerns analyzing the individual "parts" of the system; system level validation concerns analyzing the operation of the system as a whole.

Each component in P$^2$CySeMoL has been validated qualitatively by literature reviews, interviews with domain experts and case studies, and quantitatively by experiments and/or surveys. For example, the effectiveness of automated network vulnerability scanners was first studied through a literature review, then by four interviews with domain experts, then using an experiment, and finally through two case studies. The validation of each component has also been addressed from various perspectives. For instance, the rule set used in the study described in paper 7 was chosen based on a longitudinal analysis on the yearly development of the Snort Sourcefire rule set. Similarly, the survey based studies described in papers 3-5 were based on best-practice recommendations regarding data collection methodology [29, 36, 43, 123] and analysis [38, 39, 75].

A Turing test was used to analyze the system level validity of CySeMoL [162]. This test involved estimating the vulnerability of a specific object model using CySeMoL and five experts. The validity of the estimates provided by each entity were then scored by two expert evaluators. In summary, CySeMoL performed worse than the best expert, better than the worst expert, and on par with an average expert. As P$^2$CySeMoL builds (and improves) upon CySeMoL, this result could be valid also for P$^2$CySeMoL. However, no new Turing test has been performed to validate such a claim.

To further study the validity of P$^2$CySeMoL on a system level, this research used the data and methodology presented in paper 1. The P$^2$CySeMoL object model of the architecture presented in paper 1 includes 101 assets, 511 attack steps, 332 defenses, 996 connections between assets, and 1425 connections between attack steps. Of especial importance to the analysis: 6 operating systems, 30 application servers, 16 software products, 6 web applications, 30 access control points, and two work-days available for the attack. While this architecture only covers a subset of P$^2$CySeMoL's scope (see Section 3.3), there is to the authors knowledge no better empirical alternative available. Consequently, it can be seen as a reasonable starting point for further empirical validation. As is described in paper 8, P$^2$CySeMoL shows a significant correlation in the expected direction ($-0.388$, $p = 0.023$). This is a stronger correlation than for any of the other 18 metrics tested by the study presented in paper 1. While promising, it should still be viewed with care due to the specific scenario and lack of samples.

Perhaps of higher merit, P$^2$CySeMoL has been tested in two real-world case studies with positive results. In the more comprehensive of these two, the experts who designed and implemented the analyzed architecture reviewed - and agreed on - the mitigation suggestions provided by P$^2$CySeMoL (in the other study, no review was made). This suggests that while the estimates produced by P$^2$CySeMoL likely are not perfect, they prove satisfactory for supporting enterprise decision making.

# Chapter 5

# Conclusions and future work

The purpose of this thesis is to help enterprise decision makers analyze the cybersecurity, or vulnerability, of their enterprise architectures in a meaningful and understandable way. To accomplish this purpose, the attack graph model P$^2$CySeMoL was developed. P$^2$CySeMoL extends CySeMoL, a security tool that was built for the same purpose, but with a focus on SCADA systems. In summary, P$^2$CySeMoL extends CySeMoL with:

- A new modeling and analysis framework (P$^2$AMF)
- New assets, attack steps and defenses
- Possibility to model arbitrary attacker effort
- Validation and case studies

The results indicate that the contribution help to facilitate decision making of cyber security matters. They also suggest that this can be achieved with minor effort required for data collection and analysis. Thus, we argue that the main purpose of this thesis has been accomplished. Additionally, researchers can use P$^2$CySeMoL to extend existing risk-based assessment methods, for instance, as a plug-in to models that measure the potential economical losses due to successful cyber attacks [30]. The causal relations and variables in it could also aid with the design of security experiments. For instance, P$^2$CySeMoL denotes that security training has a strong correlation with the efficiency of static code analyzers in terms of reducing the vulnerability of web application. Any investigation of static code analyzers for discovering and mitigating web application vulnerabilities should thus also measure the training of the individuals using these tools.

There are however many aspects of P$^2$CySeMoL that could be further improved. The perhaps most important aspect is related to how results are presented to the end user. Currently, the user is required to manually make changes to an object model and recalculate the result in order to investigate whether one configuration is more secure than another. This can be problematic given the designated user profile of P$^2$CySeMoL - CISOs can neither be expected to have significant time to spend on such analysis, nor sufficient knowledge of IT security vulnerabilities and their dependencies to elicit key scenarios to compare [159]. To manage this property, P$^2$CySeMoL should be extended to allow automated generation and execution of different configurational scenarios, and then present the most beneficial solutions (the configurations that have the lowest vulnerability) to the end user.

A second aspect is related to the scope of P$^2$CySeMoL. There are various assets, attacks and defenses that are not available in it. In particular, it only models software exploits that

result in administrator/root privileges; whereas in practice, attackers might first exploit some vulnerability to gain user privileges, and then escalate these privileges to administrator through a second vulnerability. Future work would benefit from extending P$^2$CySeMoL to cover software attacks involving privileges other than administrator.

A third aspect is related to how data is collected to create P$^2$CySeMoL object models. While effort has been spent to make this process less time-consuming, it could still be improved. For example, pose that a user wants to depict a clean installation of the operating system Windows 7 Professional SP1. To accomplish this in P$^2$CySeMoL, the user is required to model the assets `OperatingSystem`, `SoftwareProduct`, `AccessControlPoint`, the various `ApplicationServers` and `ApplicationClients` that are installed by default, and then connect these in an appropriate means. As this configuration would be the same for any clean installation of Windows 7 Professional SP1, it means a lot of redundant work for the end user. One solution to this problem would be to create default templates for common use-cases that end users model frequently. Another means of decreasing data collection costs could be to couple P$^2$CySeMoL to data sources that contain structured information about different configurations. For instance, the US National Vulnerability Database[1] (NVD) relates information about software vulnerabilities to software products; thus, if a user inputs a software name according to the standard in NVD, this website could be queried to automatically populate the software with information about any public vulnerabilities. Similarly, the Exploit Database[2] (EDB) could be queried to investigate whether there are any public exploits for vulnerabilities elicited using NVD.

A fourth aspect is related to the attacker profile covered by P$^2$CySeMoL - professional penetration testers. Not only is this profile wide and informal, some enterprises might also perceive other threats as more critical to consider. For instance, some might prefer considering script-kiddies, adversaries with very limited technical capability who rely on readily available tools created by more experienced individuals; others might prefer modeling extremely competent adversaries such as nations or states (i.e., advanced persistent threats). To enable modeling these attacker profiles, there is a need to alter the scope of P$^2$CySeMoL's metamodel (some attacks are not applicable for certain attacker profiles), and alter the quantitative data within it.

A fifth aspect, tightly coupled to the attacker profile, is regarding the actual likelihood that different types of vulnerabilities are exploited. That is, just because a vulnerability *can* be exploited does not suggest that it actually *will* be. Allodi et al. [9, 10] compare the overlap between different datasets of software vulnerabilities to examine this property: 1) the overall number of public vulnerabilities of different severity (using NVD), 2) vulnerabilities that have public exploits (using EDB), 3) Symantec's threat databases of vulnerabilities that are exploited in the wild[3,4], 4) observations of actual data gathered in the wild by Symantec (WINE [52]), and 5) vulnerabilities actively exploited by exploit kits gathered by the authors (EKITS [9]). The authors found that few critical software vulnerabilities are exploited in practice (i.e., present in WINE or EKITS). In addition, they observed that presence of public exploits is a poor indicator regarding whether an exploit actually will show up in the wild. To increase the practical usability of P$^2$CySeMoL, vulnerabilities that currently are exploited in the wild could receive a stronger weighting factor when presenting calculation results; highlighting that these are critical to mitigate.

A sixth aspect concerns that the effectiveness of different defenses and attacks naturally vary over time, an issue that P$^2$CySeMoL's more static estimates currently fail to capture.

[1] http://nvd.nist.gov/
[2] http://www.exploit-db.com/
[3] http://www.symantec.com/security_response/attacksignatures/
[4] http://www.symantec.com/security_response/threatexplorer/

The deterioration of different theory within P$^2$CySeMoL should be studied in depth to determine when different assets, attacks, defenses, relations and quantitative data should be revised.

Finally, P$^2$CySeMoL could certainly use more thorough usability and validation tests to ascertain correctness and ease of use.

# Bibliography


[1] M. Ahmed, E. Al-Shaer, and L. Khan. A novel quantitative approach for measuring network security. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1957–1965, 2008. doi: 10.1109/INFOCOM.2008.260.

[2] H. Akaike. Factor analysis and AIC. *Psychometrika*, 52, 1987. doi: 10.1007/BF02294359.

[3] C. Alberts, A. Dorofee, J. Stevens, and C. Woody. Introduction to the octave approach. *Pittsburgh, PA, Carnegie Mellon University*, 2003.

[4] O. Alhazmi and Y. Malaiya. Modeling the vulnerability discovery process. In *Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on*, pages 10–pp. IEEE, 2005.

[5] O. Alhazmi and Y. Malaiya. Quantitative vulnerability assessment of systems software. In *Proc. Annual Reliability and Maintainability Symposium*, pages 615–620, January 2005.

[6] O. Alhazmi and Y. Malaiya. Measuring and enhancing prediction capabilities of vulnerability discovery models for apache and iis http servers. In *Software Reliability Engineering, 2006. ISSRE'06. 17th International Symposium on*, pages 343–352. IEEE, 2006.

[7] O. Alhazmi and Y. Malaiya. Application of vulnerability discovery models to major operating systems. *Reliability, IEEE Transactions on*, 57(1):14–22, 2008.

[8] O. Alhazmi, Y. Malaiya, and I. Ray. Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers & Security*, 26(3):219–228, 2007.

[9] L. Allodi and F. Massacci. A preliminary analysis of vulnerability scores for attacks in wild: the ekits and sym datasets. In *Proceedings of the 2012 ACM Workshop on Building analysis datasets and gathering experience returns for security*, pages 17–24. ACM, 2012.

[10] L. Allodi, W. Shim, and F. Massacci. Quantitative assessment of risk reduction with cybercrime black market monitoring. 2013.

[11] J. Alves-Foss and S. Barbosa. Assessing computer security vulnerability. *ACM SIGOPS Operating Systems Review*, 29(3):3–13, 1995.

[12] R. Anderson. Why information security is hard-an economic perspective. In *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pages 358–365. IEEE, 2001.

[13] R. Anderson and T. Moore. The economics of information security. *Science*, 314 (5799):610–613, 2006.

[14] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, et al. Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance. *Power Systems, IEEE Transactions on*, 20(4): 1922–1928, 2005.

[15] N. Antunes and M. Vieira. Benchmarking vulnerability detection tools for web services. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 203–210. IEEE, 2010.

[16] A. Atzeni and A. Lioy. Why to adopt a security metric? a brief survey. In *Quality of Protection*, pages 1–12. Springer, 2006.

[17] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, 2004.

[18] M. Baggett. Effectiveness of Antivirus in Detecting Metasploit Payloads . Available on `http://www.sans.org/reading_room/whitepapers/casestudies/effectiveness-antivirus-detecting-metasploit-payloads_2134`, accessed April 19, 2013, 2008.

[19] R. Böhme and F. Freiling. On metrics and measurements. In I. Eusgeld, F. Freiling, and R. Reussner, editors, *Dependability Metrics*, volume 4909 of *Lecture Notes in Computer Science*, pages 7–13. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-68946-1. doi: 10.1007/978-3-540-68947-8_2. URL `http://dx.doi.org/10.1007/978-3-540-68947-8_2`.

[20] L. Bilge and T. Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844. ACM, 2012.

[21] M. Bishop. What is computer security? *Security & Privacy, IEEE*, 1(1):67–69, 2003.

[22] M. Bishop. *Introduction to computer security*. Addison-Wesley Professional, 2004.

[23] L. D. Bodin, L. A. Gordon, and M. P. Loeb. Evaluating information security investments using the analytic hierarchy process. *Communications of the ACM*, 48(2): 78–83, 2005.

[24] W. Boyer and M. Mcqueen. Ideal based cyber security technical metrics for control systems. *Critical Information Infrastructures Security*, pages 246–260, 2008.

[25] R. Breu, F. Innerhofer-Oberperfler, and A. Yautsiukhin. Quantitative assessment of enterprise security system. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 921–928. IEEE, 2008.

[26] B. Bulgurcu. Information security policy compliance: an empirical study of rationality-based beliefs and information security awareness. *MIS Quarterly*, 34(3): 523–548.

[27] M. Buschle, H. Holm, T. Sommestad, M. Ekstedt, and K. Shahzad. A tool for automatic enterprise architecture modeling. In *IS Olympics: Information Systems in a Diverse World*, pages 1–15. Springer, 2012.

[28] M. Buschle, P. Johnson, and K. Shahzad. The enterprise architecture analysis tool–support for the predictive, probabilistic architecture modeling framework. 2013.

[29] S. Cavusgil and L. Elvey-Kirk. Mail survey response behavior: A conceptualization of motivating factors and an empirical study. *European Journal of Marketing*, 32(11/12): 1165–1192, 1998.

[30] H. Cavusoglu, B. Mishra, and S. Raghunathan. A model for evaluating it security investments. *Communications of the ACM*, 47(7):87–92, 2004.

[31] J. A. Cazier and B. D. Medlin. Password security: An empirical investigation into e-commerce passwords and their crack times. *Information Systems Security*, 15(6): 45–55, 2006.

[32] CCRA. Common Criteria for Information Technology Security Evaluation. Available on `http://www.commoncriteriaportal.org/`, accessed June 24, 2013, 2012.

[33] S. Chai, M. Kim, and H. R. Rao. Firms' information security investment decisions: Stock market evidence of investors' behavior. *Decision Support Systems*, 50(4):651 – 661, 2011. doi: http://dx.doi.org/10.1016/j.dss.2010.08.017.

[34] I. Chowdhury, B. Chan, and M. Zulkernine. Security metrics for source code structures. In *Proceedings of the fourth international workshop on Software engineering for secure systems*, pages 57–64. ACM, 2008.

[35] 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*, pages 22–33. ACM, 2010.

[36] R. Clemen and R. Winkler. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19(2):187–203, 1999.

[37] B. D. Cone, C. E. Irvine, M. F. Thompson, and T. D. Nguyen. A video game for cyber security training and awareness. *Computers & Security*, 26(1):63–72, 2007.

[38] R. M. Cooke. *Experts in Uncertainty. Opinion and Subjective Probability in Science.* Oxford University Press, 1991.

[39] R. M. Cooke and L. L. Goossens. Tu delft expert judgment data base. *Reliability Engineering & System Safety*, 93(5):657–674, 2008.

[40] J. Coughlan, M. Lycett, and R. D. Macredie. Communication issues in requirements elicitation: a content analysis of stakeholder experiences. *Information and Software Technology*, 45(8):525–537, 2003.

[41] R. E. Courtney and D. A. Gustafson. Shotgun correlations in software measures. *Software Engineering Journal*, 8(1):5–13, 1993.

[42] J. Cox. Information systems user security: A structured model of the knowing-doing gap. *Computers in Human Behavior*, 28(5):1849 – 1858, 2012. ISSN 0747-5632. doi: http://dx.doi.org/10.1016/j.chb.2012.05.003. URL `http://www.sciencedirect.com/science/article/pii/S0747563212001318`.

[43] L. Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16 (3):297–334, 1951.

[44] M. Dell'Amico, P. Michiardi, and Y. Roudier. Password strength: an empirical analysis. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[45] F. den Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen. Model-based security analysis in seven steps-a guided tour to the coras method. *BT Technology Journal*, 25(1):101–117, 2007.

[46] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[47] R. Dodge, C. Carver, and A. Ferguson. Phishing for user security awareness. *Computers & Security*, 26(1):73–80, 2007.

[48] R. C. Dodge and A. J. Ferguson. Using phishing for user email security awareness. In *Security and Privacy in Dynamic Environments*, pages 454–459. Springer, 2006.

[49] M. Dornseif and S. May. Modelling the costs and benefits of honeynets. *arXiv preprint cs/0406057*, 2004.

[50] A. Doupé, M. Cova, and G. Vigna. Why johnny can't pentest: An analysis of black-box web vulnerability scanners. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 111–131. Springer, 2010.

[51] D. P. Duggan, M. Berg, J. Dillinger, and J. Stamp. Penetration testing of industrial control systems. *Sandia National Laboratories*, 2005.

[52] T. Dumitras and D. Shou. Toward a standard benchmark for computer security research: The worldwide intelligence network environment (wine). In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pages 89–96. ACM, 2011.

[53] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.

[54] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074. ACM, 2008.

[55] I. A. Elia, J. Fonseca, and M. Vieira. Comparing sql injection detection tools using attack injection: An experimental study. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, pages 289–298. IEEE, 2010.

[56] N. Fenton. Software measurement: A necessary scientific basis. *Software Engineering, IEEE Transactions on*, 20(3):199–206, 1994.

[57] J. Fonseca, M. Vieira, and H. Madeira. Testing and comparing web vulnerability scanning tools for sql injection and xss attacks. In *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, pages 365–372. IEEE, 2007.

[58] J. Fonseca, M. Vieira, and H. Madeira. The web attacker perspective-a field study. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, pages 299–308. IEEE, 2010.

[59] M. D. Ford, K. Keefe, E. LeMay, W. H. Sanders, and C. Muehrcke. Implementing the advise security modeling formalism in mobius. In *roceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013.

[60] S. Frei, B. Tellenbach, and B. Plattner. 0-day patch exposing vendors (in) security performance. *BlackHat Europe, Amsterdam, NL*, 2008.

[61] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1300–1309. LAWRENCE ERLBAUM ASSOCIATES LTD, 1999.

[62] M. Frigault and L. Wang. Measuring network security using bayesian network-based attack graphs. In *32nd Annual IEEE International Computer Software and Applications*, pages 698–703. IEEE, 2008.

[63] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, pages 23–30. ACM, 2008.

[64] G. G. Gable. Integrating case study and survey research methods: an example in information systems. *European Journal of Information Systems*, 3(2):112–126, 1994.

[65] L. Gallon. On the impact of environmental metrics on cvss scores. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 987–992. IEEE, 2010.

[66] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.

[67] M. Goodyear, H. T. Goerdel, S. Portillo, and L. Williams. Cybersecurity management in the states: The emerging role of chief information security officers. *IBM Center for the Business of Government*, 2010.

[68] D. Hadžiosmanović, L. Simionato, D. Bolzoni, E. Zambon, and S. Etalle. N-gram against the machine: on the feasibility of the n-gram network analysis for binary protocols. In *Research in Attacks, Intrusions, and Defenses*, pages 354–373. Springer, 2012.

[69] I. Haller, A. Slowinska, M. Neugschwandtner, and H. Bos. Dowsing for overflows: a guided fuzzer to find buffer boundary violations. In *Proceedings of the 22nd USENIX conference on Security*, pages 49–64. USENIX Association, 2013.

[70] S. Hansman and R. Hunt. A taxonomy of network and computer attack methodologies. *Retrieved March*, 22:2007, 2003.

[71] H. Holm and M. Ekstedt. A metamodel for web application injection attacks and countermeasures. In *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*, pages 198–217. Springer, 2012.

[72] H. Holm, T. Sommestad, J. Almroth, and M. Persson. A quantitative evaluation of vulnerability scanning. *Information Management & Computer Security*, 19(4):231–247, 2011. doi: 10.1108/09685221111173058.

[73] H. Holm, M. Buschle, R. Lagerström, and M. Ekstedt. Automatic data collection for enterprise architecture models. *Software & Systems Modeling*, pages 1–17, 2012.

[74] 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*, 18(6):732–749, 2012.

[75] H. Holm, T. Sommestad, M. Ekstedt, and N. Honeth. Indicators of expert judgement and their significance: an empirical investigation in the area of cyber security. *Expert Systems*, pages n/a–n/a, 2013. ISSN 1468-0394. doi: 10.1111/exsy.12039. URL `http://dx.doi.org/10.1111/exsy.12039`.

[76] J. Homer and X. Ou. Sat-solving approaches to context-aware enterprise network security management. *Selected Areas in Communications, IEEE Journal on*, 27(3): 315–322, 2009.

[77] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal. Aggregating vulnerability metrics in enterprise networks using attack graphs. *Journal of Computer Security*, 21(4):561–597, 2013.

[78] S. H. Houmb, V. N. Franqueira, and E. A. Engum. Quantifying security risk level from cvss estimates of frequency and impact. *Journal of Systems and Software*, 83 (9):1622–1634, 2010.

[79] M. Howard, J. Pincus, and J. M. Wing. *Measuring relative attack surfaces*. Springer, 2005.

[80] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah. Distilling critical attack graph surface iteratively through minimum-cost sat solving. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 31–40, New York, NY, USA, 2011. ACM. doi: 10.1145/2076732.2076738.

[81] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 117–126, dec. 2009. doi: 10.1109/ACSAC.2009.21.

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

[83] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[84] S. Jajodia, S. Noel, and B. O'Berry. Topological analysis of network attack vulnerability. In *Managing Cyber Threats*, pages 247–266. Springer, 2005.

[85] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. Cauldron mission-centric cyber situational awareness with defense in depth. In *MILITARY COMMUNICATIONS CONFERENCE, 2011-MILCOM 2011*, pages 1339–1344. IEEE, 2011.

[86] W. Jansen. *Directions in security metrics research*. DIANE Publishing, 2010.

[87] N. Jegadeesh. Evidence of predictable behavior of security returns. *The Journal of Finance*, 45(3):881–898, 1990.

[88] F. V. Jensen. *An introduction to Bayesian networks*, volume 74. UCL press London, 1996.

[89] T. D. Jick. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative science quarterly*, 24(4):602–611, 1979.

[90] T. Jim, J. G. Morrisett, D. Grossman, M. W. Hicks, J. Cheney, and Y. Wang. Cyclone: A safe dialect of c. In *USENIX Annual Technical Conference, General Track*, pages 275–288, 2002.

[91] H. Joh, J. Kim, and Y. K. Malaiya. Vulnerability discovery modeling using weibull distribution. In *Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on*, pages 299–300. IEEE, 2008.

[92] F. Johansson and G. Falkman. Implementation and integration of a bayesian network for prediction of tactical intention into a ground target simulator. In *Information Fusion, 2006 9th International Conference on*, pages 1–7. IEEE, 2006.

[93] P. Johnson, J. Ullberg, M. Buschle, U. Franke, and K. Shahzad. P2amf: Predictive, probabilistic architecture modeling framework. In *International IFIP Working Conference on Enterprise Interoperability Information, Services and Processes for the Interoperable Economy and Society*, 2013.

[94] Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 27, IT Security techniques. ISO 27000. Available on `http://standards.iso.org/ittf/`, accessed June 24, 2013, 2012.

[95] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *Software Engineering, IEEE Transactions on*, 23(4): 235–245, 1997.

[96] J. Jürjens. Umlsec: Extending uml for secure systems development. In *« UML » 2002 – The Unified Modeling Language*, pages 412–425. Springer, 2002.

[97] J. Kim, Y. K. Malaiya, and I. Ray. Vulnerability discovery in multi-version software systems. In *High Assurance Systems Engineering Symposium, 2007. HASE'07. 10th IEEE*, pages 141–148. IEEE, 2007.

[98] V. Kiriansky, D. Bruening, and S. P. Amarasinghe. Secure execution via program shepherding. In *USENIX Security Symposium*, volume 92, 2002.

[99] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, et al. Factorization of a 768-bit rsa modulus. In *Advances in Cryptology–CRYPTO 2010*, pages 333–350. Springer, 2010.

[100] S. Kondakci. A concise cost analysis of internet malware. *Computers & Security*, 28 (7):648–659, 2009.

[101] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *arXiv preprint arXiv:1303.7397*, 2013.

[102] Y. Lai and P. Hsia. Using the vulnerability information of computer systems to improve the network security. *Computer Communications*, 30(9):2032–2047, 2007. ISSN 0140-3664.

[103] M. Lankhorst. *Enterprise Architecture At Work*. Springer, Heidelberg, 2005.

[104] M. Lelarge. Economics of malware: Epidemic risks model, network externalities and incentives. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 1353–1360. IEEE, 2009.

[105] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. Model-based security metrics using adversary view security evaluation (advise). In *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, pages 191–200. IEEE, 2011.

[106] D. Leversage and E. James. Estimating a system's mean time-to-compromise. *Security & Privacy, IEEE*, 6(1):52–60, 2008.

[107] D. J. Leversage and E. J. Byres. Comparing electronic battlefields: Using mean time-to-compromise as a comparative security metric. In *Computer Network Security*, pages 213–227. Springer, 2007.

[108] E. Levy. Approaching zero [attack trends]. *Security & Privacy, IEEE*, 2(4):65–66, 2004.

[109] H. Li, J. Zhang, and R. Sarathy. Understanding compliance with internet use policy from the perspective of rational choice theory. *Decision Support Systems*, 48(4):635–645, 2010.

[110] R. P. Lippmann and K. W. Ingols. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[111] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 12–26. IEEE, 2000.

[112] R. P. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[113] Q. Liu and Y. Zhang. Vrss: A new system for rating and scoring vulnerabilities. *Computer Communications*, 34(3):264–273, 2011.

[114] M. V. Mahoney and P. K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *Recent Advances in Intrusion Detection*, pages 220–237. Springer, 2003.

[115] P. K. Manadhata and J. M. Wing. An attack surface metric. *Software Engineering, IEEE Transactions on*, 37(3):371–386, 2011.

[116] G. V. Marconato, M. Kaâniche, and V. Nicomette. A vulnerability life cycle-based security modeling and evaluation approach. *The Computer Journal*, 2012. doi: 10.1093/comjnl/bxs112.

[117] S. Marechal. Advances in password cracking. *Journal in computer virology*, 4(1): 73–81, 2008.

[118] S. Massoud Amin and B. F. Wollenberg. Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41, 2005.

[119] 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*, 3(4):262–294, 2000.

[120] M. McQueen, W. Boyer, M. Flynn, and G. Beitel. Time-to-compromise model for cyber risk reduction estimation. *Quality of Protection*, pages 49–64, 2006.

[121] P. Mell, K. Scarfone, and S. Romanosky. *CVSS: A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. FIRST: Forum of Incident Response and Security Teams, jun 2007. URL `http://www.first.org/cvss/cvss-guide.html#i3`.

[122] C. Miller. The legitimate vulnerability market: the secretive world of 0-day exploit sales. In *Workshop on the Economics of Information Security (WEIS)*, pages 7–8, 2007.

[123] D. C. Montgomery, D. C. Montgomery, and D. C. Montgomery. *Design and analysis of experiments*, volume 7. Wiley New York, 1984.

[124] D. Moody. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *Software Engineering, IEEE Transactions on*, 35 (6):756–779, 2009.

[125] J. A. Morales, R. Sandhu, and S. Xu. Evaluating detection and treatment effectiveness of commercial anti-malware programs. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 31–38. IEEE, 2010.

[126] I. Nai Fovino, A. Carcano, M. Masera, and A. Trombetta. An experimental investigation of malware attacks on scada systems. *International Journal of Critical Infrastructure Protection*, 2(4):139–145, 2009.

[127] V. H. Nguyen and F. Massacci. An idea of an independent validation of vulnerability discovery models. In *Engineering Secure Software and Systems*, pages 89–96. Springer, 2012.

[128] D. Nicol, W. Sanders, and K. Trivedi. Model-based evaluation: From dependability to security. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):48–65, 2004.

[129] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, and K. Prole. Advances in topological vulnerability analysis. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 124–129. IEEE, 2009.

[130] R. M. O'Keefe and D. E. O'Leary. Expert system verification and validation: a survey and tutorial. *Artificial Intelligence Review*, 7(1):3–42, 1993.

[131] A. One. Smashing the stack for fun and profit. *Phrack magazine*, 7(49):365, 1996.

[132] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 336–345, New York, NY, USA, 2006. ACM. doi: 10.1145/1180405.1180446.

[133] A. Ozment and S. E. Schechter. Milk or wine: does software security improve with age. In *Proceedings of the 15th conference on USENIX Security Symposium*, volume 15, pages 93–104, 2006.

[134] Peter Glaskowsky. Bruce Schneier's new view on Security Theater. Available on `http://news.cnet.com/8301-13512_3-9915030-23.html`, accessed June 24, 2013, 2008.

[135] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.

[136] P. Ratanaworabhan, B. Livshits, and B. Zorn. Nozzle: A defense against heap-spraying code injection attacks. In *Proceedings of the 18th conference on USENIX security symposium*, pages 169–186. USENIX Association, 2009.

[137] R. H. Reussner, H. W. Schmidt, and I. H. Poernomo. Reliability prediction for component-based software architectures. *Journal of Systems and Software*, 66(3):241 – 252, 2003. ISSN 0164-1212. doi: http://dx.doi.org/10.1016/S0164-1212(02)00080-8. URL `http://www.sciencedirect.com/science/article/pii/S0164121202000808`. <ce:title>Software architecture – Engineering quality attributes</ce:title>.

[138] RSA Laboratories. The RSA Factoring Challenge. Available on `http://www.rsa.com/rsalabs/node.asp?id=2092`, accessed June 25, 2013, 2007.

[139] J. Ryan and D. J. Ryan. Performance metrics for information security risk management. *Security & Privacy, IEEE*, 6(5):38–44, 2008.

[140] S. E. Schechter. Quantitatively differentiating system security. In *The First Workshop on Economics and Information Security*, pages 16–17. Citeseer, 2002.

[141] B. Schneier. Cryptographic design vulnerabilities. *Computer*, 31(9):29–33, 1998.

[142] B. Schneier. Beyond Security Theater. Available on `http://www.schneier.com/essay-292.html`, accessed June 24, 2013, 2009.

[143] T. Scholte, D. Balzarotti, and E. Kirda. Have things changed now? an empirical study on input validation vulnerabilities in web applications. *Computers & Security*, 31(3):344–356, 2012.

[144] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda. An empirical analysis of input validation mechanisms in web applications and languages. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1419–1426. ACM, 2012.

[145] R. Seacord. Secure coding in c and c++ of strings and integers. *Security & Privacy, IEEE*, 4(1):74–76, 2006.

[146] 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*, pages 298–307. ACM, 2004.

[147] M. Shahzad, M. Shafiq, and A. Liu. A large scale exploratory analysis of software vulnerability life cycles. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 771 –781, june 2012. doi: 10.1109/ICSE.2012.6227141.

[148] W. F. Sharpe. Imputing expected security returns from portfolio composition. *Journal of Financial and Quantitative Analysis*, 9(03):463–472, 1974.

[149] R. Shaw, C. C. Chen, A. L. Harris, and H.-J. Huang. The impact of information richness on information security awareness training effectiveness. *Computers & Education*, 52(1):92 – 100, 2009. doi: http://dx.doi.org/10.1016/j.compedu.2008.06.011.

[150] S. Sheng, B. Magnien, P. Kumaraguru, A. Acquisti, L. Cranor, J. Hong, and E. Nunge. Anti-phishing phil: The design and evaluation of a game that teaches people not to fall for phish. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 88–99. ACM, 2007.

[151] M. Shepperd. A critique of cyclomatic complexity as a software metric. *Software Engineering Journal*, 3(2):30–36, 1988.

[152] O. Sheyner and J. Wing. Tools for generating and analyzing attack graphs. In *Formal methods for components and objects*, pages 344–371. Springer, 2004.

[153] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.

[154] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *Software Engineering, IEEE Transactions on*, 37(6):772–787, 2011.

[155] M. Siponen, S. Pahnila, and A. Mahmood. Employees' adherence to information security policies: an empirical study. In *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 133–144. Springer, 2007.

[156] T. Sommestad. *A framework and theory for cyber security assessments*. PhD thesis, KTH, 2012.

[157] T. Sommestad and J. Hallberg. A review of the theory of planned behaviour in the context of information security policy compliance. In *28th IFIP TC-11 SEC 2013 International Information Security and Privacy Conference*. ifip, Jul. 2013.

[158] T. Sommestad, H. Holm, M. Ekstedt, and N. Honeth. Quantifying the effectiveness of intrusion detection systems in operation through domain experts.

[159] T. Sommestad, M. Ekstedt, and P. Johnson. A probabilistic relational model for security risk analysis. *Computers & Security*, 29(6):659 – 679, 2010. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2010.02.002. URL `http://www.sciencedirect.com/science/article/pii/S0167404810000209`.

[160] T. Sommestad, M. Ekstedt, H. Holm, and M. Afzal. Security mistakes in information system deployment projects. *Information Management & Computer Security*, 19(2): 80–94, 2011.

[161] T. Sommestad, H. Holm, and M. Ekstedt. Estimates of success rates of denial-of-service attacks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 21–28. IEEE, 2011.

[162] T. Sommestad, M. Ekstedt, and H. Holm. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *Systems Journal, IEEE*, 7(3):363–373, Sept. 2012. ISSN 1932-8184. doi: 10.1109/JSYST.2012.2221853.

[163] T. Sommestad, H. Holm, and M. Ekstedt. Effort estimates for vulnerability discovery projects. In *45th Hawaii International Conference on System Science (HICSS)*, pages 5564–5573. IEEE, 2012.

[164] T. Sommestad, H. Holm, and M. Ekstedt. Estimates of success rates of remote arbitrary code execution attacks. *Information Management & Computer Security*, 20 (2):107–122, 2012.

[165] W. Sonnenreich, J. Albanese, and B. Stout. Return on security investment (rosi)-a practical quantitative model. *Journal of Research and Practice in Information Technology*, 38(1):45–56, 2006.

[166] S. Stasiukonis. Social engineering, the usb way. *Dark Reading*, 7, 2006.

[167] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ics) security. *NIST Special Publication*, 800(82):16–16, 2008.

[168] Suto, Larry. Analyzing the Effectiveness of Web Application Firewalls, 2011.

[169] Tenable. Nessus Vulnerability Scanner. Available on `http://www.tenable.com/products/nessus`, accessed June 26, 2013, 2013.

[170] M. Thomson and R. Von Solms. Information security awareness: educating your users effectively. *Information Management & Computer Security*, 6(4):167–173, 1998.

[171] C. Tichenor. A model to quantify the return on investment of information assurance. *The DISAM Journal of International Security Assistance Management*, 29(3):125–134, 2007.

[172] Trustworthy Computing Security group. Attack Surface Analyzer. Available on `http://www.microsoft.com/en-us/download/details.aspx?id=24487`, accessed June 27, 2013, 2012.

[173] M. Tupper and A. Zincir-Heywood. Vea-bility security metric: A network security analysis tool. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 950–957. IEEE, 2008.

[174] O. Uml. 2.0 OCL Specification. *OMG Adopted Specification (ptc/03-10-14)*, 2003.

[175] A. D. Veiga and J. Eloff. A framework and assessment instrument for information security culture. *Computers & Security*, 29(2):196 – 207, 2010. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/j.cose.2009.09.002. URL `http://www.sciencedirect.com/science/article/pii/S0167404809000923`.

[176] V. Verendel. Quantified security is a weak hypothesis: a critical survey of results and assumptions. In *Proceedings of the 2009 workshop on New security paradigms workshop*, pages 37–50. ACM, 2009.

[177] I. Vessey, V. Ramesh, and R. L. Glass. Research in information systems: An empirical study of diversity in the discipline and its journals. *Journal of Management Information Systems*, 19(2):129–174, 2002.

[178] A. J. A. Wang. Information security models and metrics. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 178–184. ACM, 2005.

[179] J. A. Wang, F. Zhang, and M. Xia. Temporal metrics for software vulnerabilities. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, number 44. ACM, 2008.

[180] L. Wang, A. Singhal, and S. Jajodia. Toward measuring network security using attack graphs. In *Proceedings of the 2007 ACM workshop on Quality of protection*, pages 49–54. ACM, 2007.

[181] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel. k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 2013.

[182] White Phosphorus. White Phosphorus Exploit Pack Sayonara ASLR DEP Bypass Technique. Available on `http://www.whitephosphorus.org/sayonara.txt`, accessed June 26, 2013, 2011.

[183] J. Wilander, N. Nikiforakis, Y. Younan, M. Kamkar, and W. Joosen. Ripe: runtime intrusion prevention evaluator. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 41–50. ACM, 2011.

[184] L. Williams, R. Lippmann, and K. Ingols. *GARNET: A graphical attack graph and reachability network evaluation tool*, volume 5210. Springer, 2008.

[185] S.-W. Woo, O. H. Alhazmi, and Y. K. Malaiya. An analysis of the vulnerability discovery process in web browsers. *Proc. of 10th IASTED SEA*, 6:13–15, 2006.

[186] A. Wool. A quantitative study of firewall configuration errors. *Computer*, 37(6):62–67, 2004.

[187] M. Workman. A test of interventions for security threats from social engineering. *Information Management & Computer Security*, 16(5):463–483, 2008.

[188] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610. ACM, 2006.

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

# Part II

# Papers 1 to 8

## Paper 1

# Empirical Analysis of System-Level Vulnerability Metrics through Actual Attacks

## Paper 2

# A Large-Scale Study of the Time Required to Compromise a Computer System

## Paper 3

# Effort Estimates for Web Application Vulnerability Discovery

## Paper 4

# Estimates on the Effectiveness of Web Application Firewalls Against Targeted Attacks

## Paper 5

# A Bayesian Model for Likelihood Estimations of Acquirement of Critical Software Vulnerabilities and Exploits

## Paper 6

# Performance of Automated Network Vulnerability Scanning at Remediating Security Issues

## Paper 7

# Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter?

## Paper 8

# P$^2$CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language