The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures

Teodor Sommestad, Mathias Ekstedt, and Hannes Holm

***Abstract*—The Cyber Security Modeling Language (CySeMoL) is a modeling language for enterprise-level system architectures coupled to a probabilistic inference engine. If the computer systems of an enterprise are modeled with CySeMoL, this inference engine can assess the probability that attacks on the systems will succeed. The theory used for the attack-probability calculations in CySeMoL is a compilation of research results on a number of security domains and covers a range of attacks and countermeasures. The theory has previously been validated on a component level. In this paper, the theory is also validated on a system level. A test indicates that the reasonableness and correctness of CySeMoL assessments compare with the reasonableness and correctness of the assessments of a security professional. CySeMoL's utility has been tested in case studies.**

I. Introduction

Security issues related to information technology (IT) continue to be a concern in today's society. The IT environments of many enterprises are composed of a large number of systems connected to form a complex system-of-systems. Security is also a complex problem that is difficult to master. To fully estimate the security of an enterprise's system architecture, a large number of issues must be considered. Enterprise systems security managers must be able to assess how the vulnerabilities in one system influence the vulnerabilities in other systems. In addition, security managers must be able to assess how individual vulnerabilities influence the security of the entire system-of-systems, given the protection solutions that are used in different locations in the architecture.

Enterprise systems security managers typically have a basic understanding of their organization's architecture and systems and the losses incurred if assets are compromised. However, the managers' understanding of how vulnerabilities depend on each other in the system-of-systems and how the vulnerabilities can be exploited is often hazy. Support from security theory can be obtained from security experts and the literature. However, consulting security experts and studying the literature is both costly and time-consuming. Generally, support is missing for informed decision making concerning security on the system-of-systems level.

Tools that help system-security managers to assess how vulnerabilities in one system influence the vulnerabilities of other systems in enterprise system architecture are valuable, particularly if these tools can offer support without requiring input data that are difficult to collect.

A. Contribution

This paper presents an analysis tool called the Cyber Security Modeling Language (CySeMoL). This tool is built on the framework presented in [1] and uses a probabilistic relational model (PRM) [2] to support system-security managers in security analysis. If an object model of the system architecture is created according to a predefined class model, the tool can approximate the probability that an attacker will succeed with different attacks against the system. Security expertise is not required to create the object model because the PRM specifies a theory on how attributes in the object model depend on each other. The users must only model their system architecture and properties.

The theory used in CySeMoL is based on logical relations, experimental research in the security domain, and domain experts' judgment. CySeMoL covers a variety of attacks, including software exploits, flooding attacks, abuse of obtained privileges, and social-engineering attacks. Emphasis has been placed on supporting security managers concerned with attacks on industrial control systems (also known as Supervisory Control and Data Acquisition (SCADA) systems). However, the tool can be used for other types of domain.

This paper presents CySeMoL's PRM and the validation of this PRM. The PRM has been validated on the component and system levels. On the component level, the variables and relationships have been validated using the literature and domain experts. On the system level, the content validity has been tested by comparing the PRM's output with the responses of five security experts to a number of scenarios. In addition, the usability of the tool is demonstrated in two case studies at large enterprises.

B. Outline

Section II presents related works. Section III briefly describes the framework presented in [1], on which this tool is built. Section IV presents the method used to create the tool. Section V presents CySeMoL's PRM. Section VI presents the results of validity tests. Section VI summarizes the paper and discusses future work.

II. Structured methods for security assessment

A substantial number of methods have been developed to quantify security and to support decision making related to security. For instance, Verendel [3] reviewed more than 100 methods for security metrication. The review presented below will cover only a subset of these methods. Emphasis is placed on methods that are applicable to assessing the security of a system-of-systems.

A number of prominent assessment methods require that the user is a security expert. For instance, the IEEE standard 27000-4 [4] and NIST's security metric guide [5] are methods that describe how an organization should develop and maintain a measurement program. However, the methods do not prescribe the measurements that should be taken or explain what different measurement values mean for security. These methods can be used as support when the security of a system-of-systems is assessed. However, they leave a substantial amount of effort to the user.

A number of methods offer concrete support and give the user a finished aggregation framework for security properties. Examples include the following: attack trees [6], defense trees [7], Boolean Logic Driven Markov Processes [8], the CORAS framework [9], Secure Tropos [10], and the model proposed by Breu et al. [11]. These methods help users combine variables to produce a meaningful result. Thus, the methods can help to combine the security values of single systems into a single value for a system-of-systems (i.e., the total risk). However, the methods require the user to produce the security ratings. For example, for defense trees, the user must quantify the likelihood of attacks being successful; for Boolean Logic Driven Markov Processes, provide time-to-compromise estimates; and in the model of Breu et al. give threat-realization probabilities. While some methodological support is available for quantification (e.g., [12]), expertise is still required. In addition, many of these methods require the users to identify causal dependencies in their systems, e.g., attack trees must be created before they can be used. For some systems, such causal models can be found in the literature, for example, the model employed [13] for electronic voting systems.

This paper describes a method that does not require security expertise from the user. In other words, the user of the method must only input information about the system architecture and is not required to provide security properties such as time-to-compromise or attack-success probability. Instead, the security properties for the system are derived from the system architecture and quantified according to a generic theory.

The practical utility of a method that quantifies the security of system architectures without requiring security expertise from the user is obvious. However, few methods of this type exist that are applicable to assessing the security of a system-of-systems. For instance, the Common Vulnerability Scoring System [14] produces assessments for a single software vulnerability, and the model presented in [15] produces assessments for single hosts.

Several of the methods that have a high abstraction level use best-practice standards to produce a security rating by the organization's compliance to the standard (e.g., Johansson's method [16]). The scope of such methods is useful where a system-of-systems is assessed. However, the ratings obtained are difficult to interpret and therefore not straightforward for system-security managers. For instance, knowing how high a value should be is difficult, as is deriving which risk is associated with a certain rating. Additionally, cause-and-effect relationships are not clear in these methods.

In recent years, a substantial number of articles have been published to develop methods that use attack graphs. An attack graph aims at determining which attacks can be conducted against a system. Because potential attacks are the source of cyber security risk, these methods fit well with decision-making processes concerning security. CySeMoL's approach is similar to the approach used in attack graphs.

Methods based on attack graphs are based on a model of the system architecture and a database of security exploits or security vulnerabilities [17], [18]. From these data, an algorithm calculates privileges and network states that can be reached by an attacker who starts from a given position [17].

Since the early variants of attack graphs appeared ([19],[20]), several tools have been developed that offer different solutions to the problem. Differences can be seen both in terms of the data they require as input and the output the produce when they are applied. The most mature tools are NETSPA [21], [22], MulVAL [23], and the TVA-tool [24]. These tools are described below.

NETSPA has been used to analyze networks of thousands of hosts, and its usability has been demonstrated in case studies [22]. However, NETSPA uses a coarse model of the attacker's capabilities. All software vulnerabilities in the database are considered to be exploitable by the attacker (given that the software can be reached) [21]. No differentiation is made with respect to the security measures implemented on the targeted host, to whether exploiting the vulnerability requires a particular configuration, or to the attacker's competence. GARNET [25] and its successor NAVIGATOR [26] build on NETSPA and add new visualization capabilities and support for what-if analysis.

MulVAL does not treat all vulnerabilities as unquestionably exploitable by the attacker. In MulVAL, each vulnerability is associated with a probability to represent how likely an attacker is to exploit the vulnerability [23]. This approach makes the model of potential attacks more expressive. Unfortunately, such probability values are not available in

vulnerability databases. In descriptions of the method, the access-complexity rate from the US National Vulnerability Database has been translated into probability values [27]. However, no arguments are given for why this translation is used, and the validity of the translation remains unclear. Additionally, the probabilities only represent success rates generally and do not take into account protective measures that increase the difficulty of exploiting a vulnerability.

The TVA tool [24] uses a database of exploits possessed by the attacker instead of a database of vulnerabilities. The exploits are associated with detailed pre- and post-conditions, which state when the exploit can be applied and what state is reached after the exploit has been applied. Thus, the analysis can be constructed to represent an attacker armed with certain exploits. However, no database of exploits exists that is described this way. The data must be entered before use.

These three attack-graph methods offer different solutions to the problem of assessing the security of a given system specification. An issue all methods must address is the complex graphs that are produced when systems of realistic sizes are analyzed. Additionally, they need to manage cycles in the graphs. Another issue is obtaining the input data. All three tools described above use the vulnerability scanner Nessus to collect these data. However, a recent accuracy test shows that Nessus misses more than half of the vulnerabilities when given access credentials to the hosts in a network and four out of five vulnerabilities when credentials are not given [28]. Thus, the automated scans on which the three tools rely are not reliable when individual vulnerabilities must be detected. In addition, in environments with sensitive systems (e.g., SCADA systems), scanners must be avoided because scanners can interrupt critical system services [29].

Another drawback of existing tools is the type of attacks that they cover. The tools are developed for software exploits targeting services running on the listening ports of machines. Thus, they lack the capability to model many relevant types of attack, e.g., password cracking, social engineering, and denial-of-service attacks. NETSPA has been extended to include attacks on clients (e.g., web browsers) [30]. However, the other two tools have not. Another matter falling outside the scope of these tools is zero-day exploits, i.e., attacks using vulnerabilities that are unknown to the public. The user of the tool can of course enter hypothetical data into the database and perform the analysis with these data. However, competence is required to identify which zero-day attacks can be reasonably expected from the attacker.

CySeMoL also models attacks and assesses the attacks that an attacker can execute. Compared with the three tools discussed above, CySeMoL analyzes a wider range of attack types and security measures. CySeMoL's output is probabilistic (as in MulVAL) and estimates the probability that different attacks can be accomplished against assets in the system architecture. The probabilities used in these calculations have been derived from experimental studies and studies eliciting the judgment of domain experts.

III. The Framework used: the PRM Template

The Cyber Security Modeling Language (CySeMoL) is built on the framework presented in [1]. This framework is a template for a probabilistic relational model (PRM) for security-risk analysis. Section III.A briefly describes the PRM formalism. Section III.B describes the security template for the PRMs presented. Section III.C describes the part of this template that is used in CySeMoL.

A. Probabilistic relational models

A PRM [2] specifies how a Bayesian network [31] should be constructed from an object model, i.e., how a Bayesian network should be created from a model that instantiates a class diagram, such as the one of the Unified Modeling Language (UML).

In a PRM, classes can have attributes and reference slots. The attributes are random variables with states from a discrete domain. The reference slots refer to other classes and express which relationships a class has with other classes. For instance, the attributes *System.Available* and *Person.Certified* could have the domain of values {True,False}. The reference slot *System.Administrator* could refer to the class *Person*.

The attributes in the PRM are associated with a set of parents. The parents of an attribute *A* are attributes in the object model that *A*'s value depends on. Parents are defined using a chain of reference slots that leads from the child attribute's object to the parent attribute's object. For instance, the attribute *System.Available* could be assigned the parent attribute *System.Administrator.Certified* using the reference slot *Administrator* of the class *System*. In this case, the slot chain is the single slot *System.Administrator*. Slot chains comprising multiple reference slots are also possible. If a slot chain points to attributes of more than one object in an instantiated model, an aggregate is used, e.g., one of the Boolean operators *OR* or *AND*.

Each attribute is associated with a conditional probability table that defines the attribute's value given all possible combinations of states in the attribute's parents. For instance, the attribute *System.Available* would be given different probabilities that express the attribute's value when *System.Administrator.Certified* is *True* and *False*. The probabilistic model enables the value of attributes in an instantiated object model to be inferred. Such inference can also infer values for attributes with no assigned state.

In essence, a PRM defines how a Bayesian network shall be generated using the attributes of an object model. Thus, a PRM constitutes a formal machinery for calculating the probabilities of object properties in various architecture instantiations. For example, a PRM could be used to assess the availability of systems given that certain administrators are assigned to the systems.

B. The PRM template for security-risk analysis

In [1], a template for PRMs is defined for security-risk analysis. This template defines abstract classes together with attributes, reference slots and conceptual-attribute parents. The

classes in this template are: *Asset*, *Owner*, *Threat*, *ThreatAgent*, *AttackStep*, and five types of *Countermeasure*. The countermeasures are: *ContingencyCountermeasure*, *PreventiveCountermeasure*, *DetectiveCountermeasure*, *ReactiveCountermeasure* and *AccountabilityCountermeasure*.

If a PRM is constructed according to this template and the PRM's conditional probabilities are assigned, the PRM can be used to perform two types of analysis. The first and more extensive analysis requires the instantiation of all the classes and can produce values for the expected economic losses for the architecture. This analysis includes consideration of the probability that different attack scenarios will be attempted and the expected loss that would be incurred if an attack is successful. The second analysis uses a subset of the template and can be used to calculate reachability values for different attack paths (threat instances), as in attack graphs. CySeMoL employs the second type of analysis.

C. The scope of the PRM

CySeMoL focuses on assessing the probability that attack paths can be accomplished given that they are attempted. Thus, CySeMoL uses a subset of the classes, attributes and dependencies defined in [1]. The class *AttackStep* is used to represent attacks together with the probability that the attacks are successful and that they are detected when they are attempted. The classes *PreventiveCountermeasure*, *DetectiveCountermeasure* and *ReactiveCountermeasure* are also included. Only one type of *ThreatAgent* is considered: a threat agent who has one week and publicly available tools.

CySeMoL's PRM includes attacks and countermeasures of relevance to industrial control and SCADA systems security. Threats against these systems are primarily related to the systems' availability and integrity properties (and not confidentiality). However, SCADA systems comprise the same type of subsystems as other information systems. Thus, the PRM can be used to analyze such systems but with limited support for threats against confidentiality. In addition, the PRM has other limitations, e.g., the countermeasures that the PRM can cover. These limitations are discussed in section IV.

IV. The Method used to construct CySeMoL

This section presents the method used to construct CySeMoL, including a description of the qualitative structure (section IV.A) and the quantitative parameters associated with this structure (section IV.B). A summary is given in section IV.C.

Because this tool has been the subject of a considerable number of studies in the security field, this section does not describe each study in detail. The interested reader can find more information about these studies in the references.

A. Qualitative structure

The PRM's qualitative structure includes everything but the quantitative parameters, i.e., the classes, reference slots, attributes, and parents of attributes. CySeMoL's qualitative structure has been developed using the literature and judgment of security experts.

1) Literature study

The literature was studied extensively to identify an initial set of assets and which attack steps to include. This research included a review of a large number of textbooks (e.g., [32]), standards and reports (e.g., [29]), overviews (e.g., [33]) and security databases (e.g., [34]). Descriptions of attacks and countermeasures in these sources were used to create an initial model of a suitable level of abstraction and scope.

When the initial model was finished, the literature on specific attacks was consulted. This literature was used to determine the parents of attack steps, i.e., the countermeasures and privileges (completed attack steps) that influenced the probability that an attack step could be accomplished. A large number of sources were used for each type of attack. For instance, sources such as [35–39] were used to identify parents of remote code-exploitation attacks, and sources such as [40–42] were used to identify the parents of password-cracking attacks.

2) Review by domain experts

Before more detailed studies were conducted on specific attack types, the initial model was reviewed by three domain experts. All three were professional penetration testers. In interviews, these domain experts were asked to evaluate whether the model included the variables that are most useful when the security of a system-of-systems is to be assessed. To be useful, a variable should not only be important to security but also possible to assess. Thus, the experts were asked to consider which information was worthwhile to collect from a decision-making perspective. Several minor changes were made based on the suggestions by the experts. For example, firewall was modeled with fewer attributes and password protection was emphasized.

In addition to the general validation of the model, a number of domain experts were consulted on specific attacks and the parents to include for these attacks. Because such input requires a good understanding of the specific type of attack, different domain experts were used for different areas (cf. Table I). For example, three persons were interviewed about intrusion-detection systems and the performance variables of such systems, and three persons were interviewed regarding remote arbitrary-code exploits. Few changes were suggested to the initial model during these validation efforts. Input from the domain experts helped to define variables in a practical way and to determine which variables to include (e.g., variables influencing the effort required to find new software vulnerabilities). Overall, the domain experts agreed with each other about variables relevance. This agreement suggests that the final model offered a good tradeoff between scope and usability.

B. Quantitative parameters

A PRM requires quantitative parameters for conditional probability distributions. CySeMoL's PRM consists of both logical dependencies with deterministic influences and

probabilistic dependencies with uncertain influence. These dependencies are used to estimate the probability that a professional penetration tester can succeed with attacks against the architecture within one week using publicly available tools.

1) Logical, deterministic dependencies

A substantial portion (82 %) of the entries in the PRM's conditional probability tables is deterministic. These deterministic dependencies are used in the following cases:

a) Attack steps that are specializations of the same goal are aggregated into one attack step to simplify the model.
b) Certain preconditions are required for an attack to be possible.

The deterministic dependencies created for the first case are modeling constructs added for practical reasons. For example, denial-of-service attacks against services are decomposed into two variables representing two ways denial-of-service attacks can be conducted (semantic attacks or flooding attacks). This decomposition makes the conditional probabilities for each attack type easier to follow.

Deterministic dependencies of the second type are present when a condition is necessary to accomplish an attack step. For example, to perform remote code execution against a software service, two necessary (but not sufficient) conditions are that the attacker must be able to send data to the port the service listens to and that the service has a software vulnerability. The second type of deterministic dependency was identified from the literature and validated by domain experts in interviews. Examples of such dependencies are given in section V.B.

2) Probabilistic, uncertain dependencies

Dependencies not determined by logical dependencies are uncertain and are defined using probabilities. Such dependencies are uncertain because the PRM does not include all the details that determine the variable's actual, which is the case if the PRM lacks a variable that could be important (e.g., the countermeasure application whitelisting) or if a variable's states represent a range of values (e.g., the severity rating of software vulnerabilities, divided into three levels). Such simplifications arise from the practical reasons discussed above, i.e., the creation of an instance model should not be costly.

Some probabilistic relationships could be specified based on published data from experiments and observations. For instance, data on the success of password cracking given different conditions could be found in [40–43]. However, for most of the conditional probabilities required, reliable quantitative data cannot be found in the literature. For instance, experiments on intrusion-detection systems are difficult to generalize from [44], and data on efforts required to find new software vulnerabilities are not gathered in a systematic way [45].

When reliable data could not be found in the literature, estimates were collected from domain experts. The data collected this way come from five surveys. The number of respondents to these surveys varies between four and 165 individuals. In four of the five surveys ([46–49]), the respondents' judgment was weighted using Cooke's classical method [50], a well-established method for weighting domain experts based on their ability to accurately assess a set of test questions on the same topic as the real questions. The effectiveness of the method is demonstrated in [50]. In the fifth study [51], the experts were weighted based on the number of real systems on which they had tested the variable's state.

C. Summary

The attacks and countermeasures included in CySeMoL were identified using the literature and input from domain experts. The aim of the qualitative structure was to be as complete as possible while remaining useful to a typical system-security manager. The quantitative parameters in the PRM are deterministic dependencies between attributes and uncertain dependencies between attributes. The probabilities for the uncertain dependencies are derived from observations of systems, experiment, and studies based on structured expert elicitation. An overview is given in Table I.

TABLE I
OVERVIEW OF METHODS USED

Part of the PRM	Qualitative validation method	Parameterization method
Discovering new vulnerabilities	Literature and 3 experts.	Cooke's classical method applied to 17 domain experts' judgment [46].
Remote arbitrary code exploitation attacks	Literature, pilot study, and 3 experts.	Cooke's classical method applied to 21 domain experts' judgment [47].
Intrusion detection	Literature and 3 experts.	Cooke's classical method applied to 165 domain experts' judgment [49].
Denial-of-service attacks	Literature and 2 experts.	Cooke's classical method applied to 23 domain experts' judgment [48].
Exploitation of network configuration mistakes	Literature and 2 domain experts.	Data described in [52] and [53] combined with four domain experts' judgment [51].
Attacks on password protection	Literature and one domain expert.	A review and synthesis of password-guessing data [40–42] and the capabilities of rainbow tables [43].
Social-engineering attacks	Literature.	Experiments [54–57] on social-engineering attacks.

V. THE CYBER SECURITY MODELING LANGUAGE'S PRM

This section describes the main contribution of this paper: CySeMoL. This section gives an overview of the metamodel (section V.A), the deterministic and probabilistic dependencies embedded in the PRM (section V.B), and the instantiation of attack paths (section V.C). A full description of all concepts and dependencies is not given here because of the space the description would require. The description presented here gives an overview of CySeMoL and provides concrete examples of parts of the model. The interested reader can download the PRM and the software tool in which the PRM is implemented from [58].

A. Metamodel overview

The metamodel comprises 22 classes, 102 attributes, and 32 class relationships (reference slots). These classes, attributes, and class relationships dictate what information an architecture model should contain and are depicted in Fig. 1. Two types of attribute are distinguished in Fig. 1: countermeasures and attack steps. The upper box in a class describes the countermeasures associated with the class. The lower box describes the attack steps associated with the class. Relationships are marked by the dashed lines between classes.

The metamodel contains three concretized types of software: *OperatingSystem*, *Service*, and *ApplicationClient*. All types of *SoftwareInstance* are related to the *SoftwareProduct* the types are an installation of. An *OperatingSystem* can be related to a *NetworkZone* in which traffic between software instances is permitted (i.e., not filtered). The class *NetworkInterface* can connect multiple instances of *NetworkZone* and mark the instances as trusted or untrusted zones. A *NetworkInterface* can allow certain instances of *DataFlow*. The other data flows are assumed to be blocked. A *DataFlow* has a *Protocol*, and a *DataFlow* can read or write to a *DataStore* owned by a *SoftwareInstance*.

The *NetworkInterface* is related to the *Firewall* that enforces the *NetworkInterface*'s rules. A *Firewall* can be related to the class *DeepPacketInspection* and to an *IDSsensor* that enhances the *Firewall*'s capabilities. An *IDSsensor* can be associated with an *OperatingSystem* when the *OperatingSystem* is a host-based intrusion-detection system. A *DeepPacketInspection* can be associated with the *Service* on which the *DeepPacketInspection* focuses or the *ApplicationClient* for which it acts as a proxy.

All types of *SoftwareInstance* can be associated with an *AccessControlPoint*, which controls access to the software, e.g., a network *Service* or *OperatingSystem*. An *AccessControlPoint* is associated with an *AuthenticationMechanism*, which authenticates access requests and the *Account* instances that are allowed access. Because passwords are common, the two preceding classes have been specialized to *PasswordAuthenticationMechanism* and *PasswordAccount*. An *Account* is owned by a *Person*, and a *Person* can be covered by an *AwarenessProgram*.

A *ZoneManagementProcess* is related to a *NetworkZone* and describes how systems within the *NetworkZone* are managed, e.g., if the machines have been hardened.

B. Attribute dependencies

The 22 classes have 102 attributes. The attributes' values in an instantiated model are used to assess the security of the modeled enterprise systems. More precisely, the probability that certain attack paths (i.e., chains of attack steps) can be accomplished is used to assess the security of the architecture. This section gives examples of attack paths that may be instantiated in an instance model and the countermeasures that influence their probability of success.

If an attacker attempts to log on to a *SoftwareInstance*, the attacker may be required to bypass an *AccessControlPoint*, i.e., if the *SoftwareInstance* has a relationship to the *AccessControlPoint*. An *AccessControlPoint* is associated with an *AuthenticationMechanism* and *Account*s that belong to *Person*s who are authorized to access the system. An *Account*'s password may be compromised by being guessed online, offline, or through social engineering.

Because password authentication is widely used, CySeMoL focuses on password authentication. The difficulty of compromising a *PasswordAccount* online depends on the presence of default passwords, whether password policies are automatically enforced, and whether the number of guesses that can be produced is limited.

The success of offline guessing depends on the possibility of extracting the password repository, whether password policies are automatically enforced, whether passwords are hashed, and whether passwords are salted. The success probability for social engineering is decreased if the account-owner is included in an *AwarenessProgram*.

Another way to obtain access to the *OperatingSystem* is to execute arbitrary code via *Services*, *ApplicationClients*, or services in the *OperatingSystem* that are unknown to the system owner. To accomplish these attack steps, the attacker must be able to connect to the *SoftwareInstance* in question, e.g., the attacked *Service* in question. Once connected, the attacker must accomplish a remote arbitrary-code attack.

To connect from another *NetworkZone*, the attacker must have access to a machine in the other *NetworkZone* and be able to send data over the *NetworkZone* that connects the two. Data can be sent if the attacker can produce a request in a *DataFlow* that has the *Service* as server or a response in a *DataFlow* that has the *ApplicationClient* as client. Requests and responses can be produced if the attacker has gained access to the software that produces the responses (e.g., an operating system) or by compromising the *DataFlow*'s integrity in a different way (e.g., by executing a man-in-the-middle attack using ARP-spoofing). To exploit unknown services in an *OperatingSystem*, the attacker must find such services. The probability that the attacker can find such services is influenced by attributes in the *ZoneManagementProcess* associated to the *OperatingSystem*'s *NetworkZone*.

To connect from the same zone, the attacker must obtain an own address in the *NetworkZone*. An address can be obtained by gaining access to an *OperatingSystem* in the zone, by breaking the *PhysicalZone* of the *NetworkZone* and connecting an own machine, or by finding an unknown entry point (e.g., an undocumented dual-homed computer) to the *NetworkZone.* The attributes in the *ZoneManagementProcess* influence the possibility the attacker may find unknown entries to a *NetworkZone.*

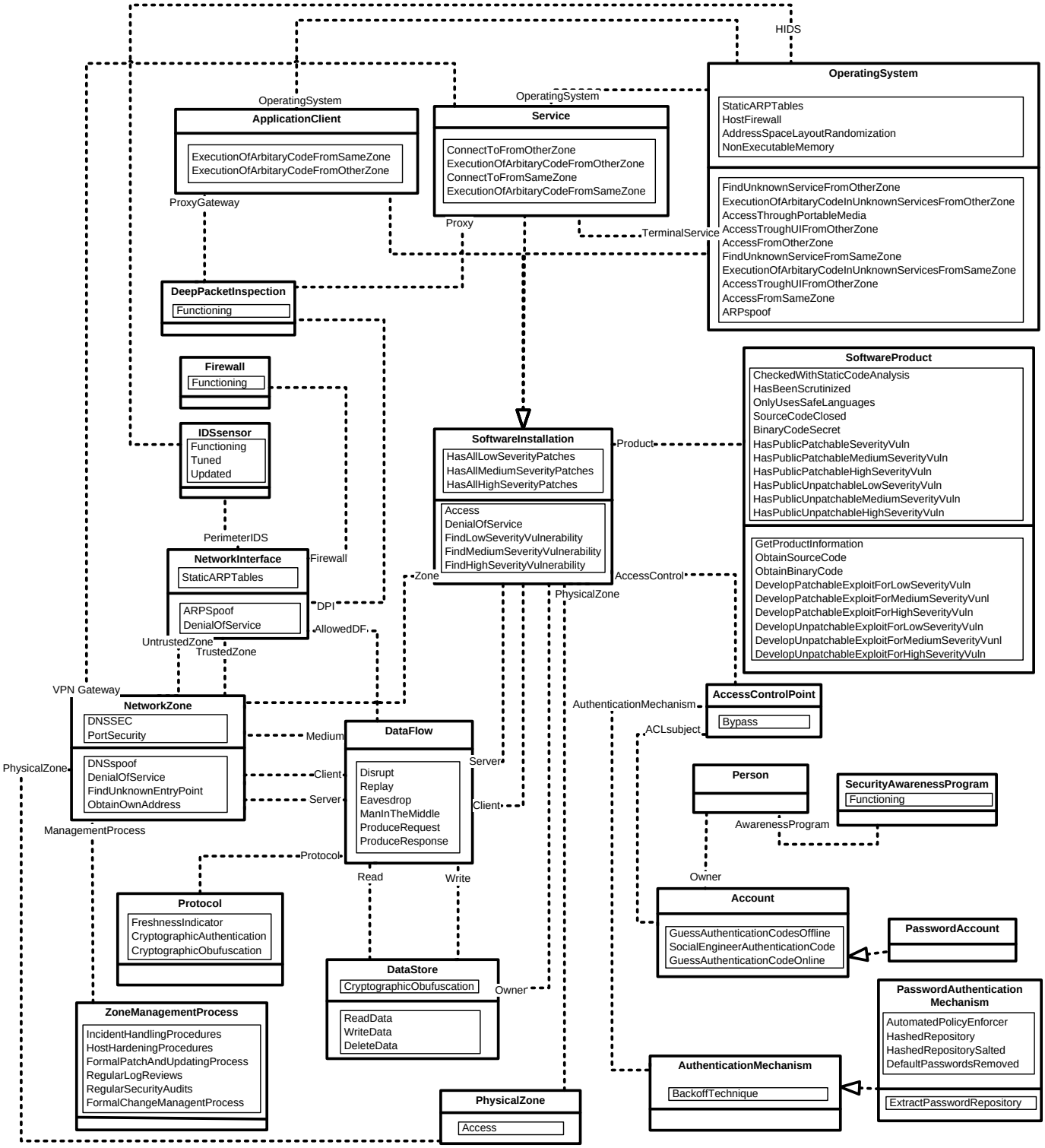

Fig. 1. Metamodel of CySeMoL's PRM. The upper box in a class contains the countermeasures associated with the class. The lower box contains the attack steps associated with the class.

When an attacker can send data to the software, the attacker can attempt to execute arbitrary code remotely. The possibility of succeeding with this approach is influenced by the presence of address space layout randomization, non-executable memory protection, and whether the attacker has access rights to the software in question. If the attack is executed from another zone, the existence of deep-packet inspection in firewalls will have an influence. An *IDSsensor* will influence the possibility of detecting the attack. The influence of the *IDSsensor* depends on whether the attack is from the same *NetworkZone* or not.

The possibility of performing remote arbitrary-code exploits is also contingent on the existence of a high-severity software vulnerability in the target. The attacker may search the

SoftwareInstance's *SoftwareProduct* for publicly known vulnerabilities that have not been patched, or the attacker may invest time in searching for previously unknown vulnerabilities in the *SoftwareProduct*. The success rate of the former approach depends on the existence of known vulnerabilities and patches as well as the patching procedures in the *ZoneManagementProcess*. The success of the latter approach depends on attributes related to the *SoftwareProduct* (e.g., if developers have tested the *SoftwareProduct's* security using static analysis tools).

A denial-of-service attack against a *SoftwareInstance* can be accomplished if the attacker has access to the *SoftwareInstance's OperatingSystem*. Network-based denial-of-service attacks can be performed against a *Service* or *OperatingSystem* if the attacker can connect to the *Service* or *OperatingSystem*. Such attacks can also be conducted against a *DataFlow* if the attacker can accomplish unavailability in associated clients, servers, or mediating *NetworkZone*s.

C. Attack-path generation and assessment

An attack path is an ordered set of attack steps. Because the causal dependencies are expressed in CySeMoL's PRM, producing an exhaustive list of all attack paths that should be assessed is straightforward as long as the maximum number of steps is specified.

For each identified attack path, the corresponding Bayesian network is created and the success probabilities for all included attack steps are calculated. The attack steps in an identified attack path will be influenced by the attack steps in the path, the attack steps not in the path, and countermeasures in the system architecture. Attack steps not in the path are assigned a success probability of zero (because these steps are not attempted). All other attribute values are calculated as the PRM prescribes.

VI. Verification and Validation

CySeMoL can be viewed as an expert system that assesses attack paths in a system architecture and estimates the probability that different attack paths can be traversed by a professional penetration tester within one week. The correctness and accuracy of this estimate is essential for the practical utility of CySeMoL. This section describes the verification and validation of CySeMoL based on the terminology and recommendations of [59].

A. Verification

Verification concerns the consistency, completeness, and correctness of the software implementation of the expert system [59]. A verification procedure can either be domain dependent and check for anomalies in the system using meta-knowledge on what is typical in the domain, or the verification procedure can be domain independent and look for general anomalies and errors in the implementation [59].

Domain-independent verification has been performed through inspections of the output produced by the tool. The result produced for both fictive test cases and real architectures has been inspected. These checks ensured that all attack paths that should be created by the model are present in the output and that the model does not contain redundant attack paths. The checks also verified that changes to a system's architecture produce the results prescribed by the theory in the model.

Domain-independent verification has focused on inspecting whether the PRM implementation is consistent (e.g., in naming attributes), complete (i.e., that all attribute parents are included), has the correct weights (i.e., the conditional probabilities), and infers data correctly (i.e., attack-generation procedure). CySeMoL is implemented as an extension to the Enterprise Architecture Analysis Tool (EAAT) [60]. EAAT implements PRM-inference with the SMILE library used in Genie [61]. Thus, the probabilistic inference mechanism has been verified in other projects.

B. Validation

An expert system's validity should be assessed in relation to a criterion [59]. CySeMoL has been validated using the criterion that CySeMoL should have expertise similar to that of a security expert.

Validity tests can be performed on a component level to validate pieces of the expert system or on a system level to validate the full expert system against the criterion. CySeMoL has been validated on both levels.

On a component level, CySeMoL has been validated by domain experts in interviews and surveys. As described in Section IV, these experts have validated the dependencies in the model and the prioritizations. In other words, the experts have validated the qualitative part of the underlying theory. The quantitative part of the theory has been validated on a component level in the studies from which the theory is developed. CySeMoL's theory is drawn from the experts directly or from published empirical studies in the domain. Thus, further tests on a component level of the quantitative model's validity by experts would be redundant.

To test the validity of CySeMoL on a system level, a Turing test was performed. Turing tests are particularly useful when the answers to test cases are unknown (or costly to determine) and it cannot be assumed that a particular domain expert is correct [59]. The Turing test was designed to validate the attack paths and estimates against the criterion, i.e., that CySeMoL performs as a domain expert. Turing tests of expert systems have several advantages over other tests [59]. However, no standards have been established for how the Turing tests should be designed. The test of CySeMoL was similar to the tests described in [68] and [71]. Two pools of human experts are used: one that produces assessments of the same type as the expert system and one that rates the first pool's assessments and the expert system's assessments based on how reasonable the assessments are. The test's design is described below.

Three system architectures were presented to five domain experts experienced in penetration testing. The system architectures were depicted in a graphical format together with

tables showing attributes of objects in the architecture during interviews lasting one hour. The graphical drawings and tables contained the information prescribed by CySeMoL's metamodel (cf. section V.A). The five domain experts were asked to reason about ways that three different attack goals could be reached in the system architecture. The experts were asked to focus on the attacks with a relatively high probability of success, i.e., to disregard attacks that are unlikely to succeed. The resulting attack scenarios contained a brief description of the attack and estimates of the probability that a professional penetration tester would succeed with the included attack steps within one week. During the hour-long interviews, the experts produced one to three attack paths or solutions for each of the nine cases presented.

To limit the time required to evaluate these solutions, a subset of the five experts' solutions was used in the Turing test. One solution from each expert was randomly selected for each of the nine cases. The same principle was applied when solutions from CySeMoL were selected: one solution was randomly selected from the three solutions with the highest probability of success. Thus, solutions to nine cases from six experts were used. One of these experts was CySeMoL. To disguise the sources of the attack scenarios, the scenarios were described using the same language and abstraction level. In addition, all probabilities were rounded to the nearest percentile, which is a factor of 5 % because this resolution was used by most of the experts.

The database of 54 solutions was then presented in randomized order in a questionnaire to two domain experts. Using a five-point scale, the experts were asked to say if they agreed with the statement "this assessment is reasonable and correct". On this scale, one means that the evaluator completely disagrees with the statement. Five means that the evaluator completely agrees with the statement.

The sample size prohibits drawing reliable statistical conclusions from this test. The median score that the evaluators gave the experts and CySeMoL attack scenarios is shown in Table II. The summary statistics indicate that the reasonableness of CySeMoL's assessments is comparable to that of the assessments of the domain experts. In mean score, CySeMoL ranked fourth in a tie with expert 2. In median score, CySeMoL ranked fifth.

TABLE II
RESULTS FROM THE TURING TEST

	Evaluator 1	Evaluator 2	Mean	Median
Expert 1	[2,4,3,2,2,2,5,4,3]	[4,4,3,4,4,2,4,4,4]	3.3	4
Expert 2	[4,4,2,2,4,2,3,2,1]	[4,4,3,3,4,2,2,4,3]	2.8	3
Expert 3	[2,4,3,4,3,3,3,4,3]	[4,2,4,5,3,4,2,4,3]	3.3	3
Expert 4	[4,1,4,2,2,3,4,4,4]	[4,2,4,3,3,3,3,4,3]	3.2	3
Expert 5	[2,2,2,1,1,1,2,2,2]	[2,2,2,2,2,2,2,2,2]	1.8	2
CySeMoL	[2,2,3,1,2,2,3,3,2]	[5,5,4,3,4,2,1,4,2]	2.8	2.5
Novice 1	[2,4,3,1,2,2,2,3,2]	[2,3,2,2,2,2,2,2,2]	2.2	2
Novice 2	[1,2,4,1,2,2,2,1,1]	[3,3,3,4,2,2,3,2,2]	2.2	2
Novice 3	[4,2,2,4,4,4,3,2,1]	[2,2,2,3,3,2,2,2,1]	2.5	2

Considerable variation exists between the evaluators' scores. A potential concern is that the scoring is arbitrary, i.e., that the experts are unable to distinguish a reasonable solution from an unreasonable one. To test the discriminatory ability of the evaluators, they were requested to evaluate the solutions of three IT security novices. These novices had more than five years of experience in information technology but without a focus on security matters. The novices' solutions were elicited the same way as the experts' solutions were elicited and their solutions were presented in the same format to the evaluators. The evaluators did not know that these solutions were produced by novices.

As shown in Table II, the novices score better individually than one expert. However, the novices receive low median and mean scores compared with the experts overall, suggesting that the evaluators can discriminate reasonable solutions from less reasonable.

C. Applicability and usability

To apply CySeMoL, the user must model the system architecture according to the metamodel depicted in Fig. 1. However, the user is not required to input all the information in the metamodel. In particular, the user does not need to input the attributes included in the lower box of the classes (the attack steps). Thus, the user is required to model concepts such as network zone, data flow, and software installation and assign values to attributes that determine whether countermeasures such as DNSSEC and non-executable memory are functioning. However, the user is not required to ascertain whether attacks succeed. In addition, for a number of attributes, the PRM can estimate values for the attributes in the upper tile. For example, the presence of unpatched publicly known vulnerabilities in installed software can be estimated based on the product's attributes and the presence of automated patching procedures.

The usability of this tool has been assessed in [63]. Areas for improvement in graphical attractiveness and the automated support for time-demanding tasks are identified in [63]. However, users without security expertise can comprehend the concepts used to model systems with CySeMoL when the textual definition of the concepts is presented.

CySeMoL has also been evaluated with respect to usability in three case studies that analyzed the security of system architectures. The case studies focused on the following: (1) the control center and adjacent environments of one of Sweden's largest electrical power utilities, (2) the electrical substations and remote communication of one of Sweden's largest power systems, and (3) reference architectures for one of the world's most common electrical-power management systems. An excerpt from an instance model with one assessed attack path is depicted in Fig. 2.

The results of these three CySeMoL applications were appreciated by the system owners, and previously unknown security issues were identified in all three studies. However, the potential for improvement in data collection and the

visualization of assessment results was identified. Meanwhile, data-collection support has been implemented (see [64]).

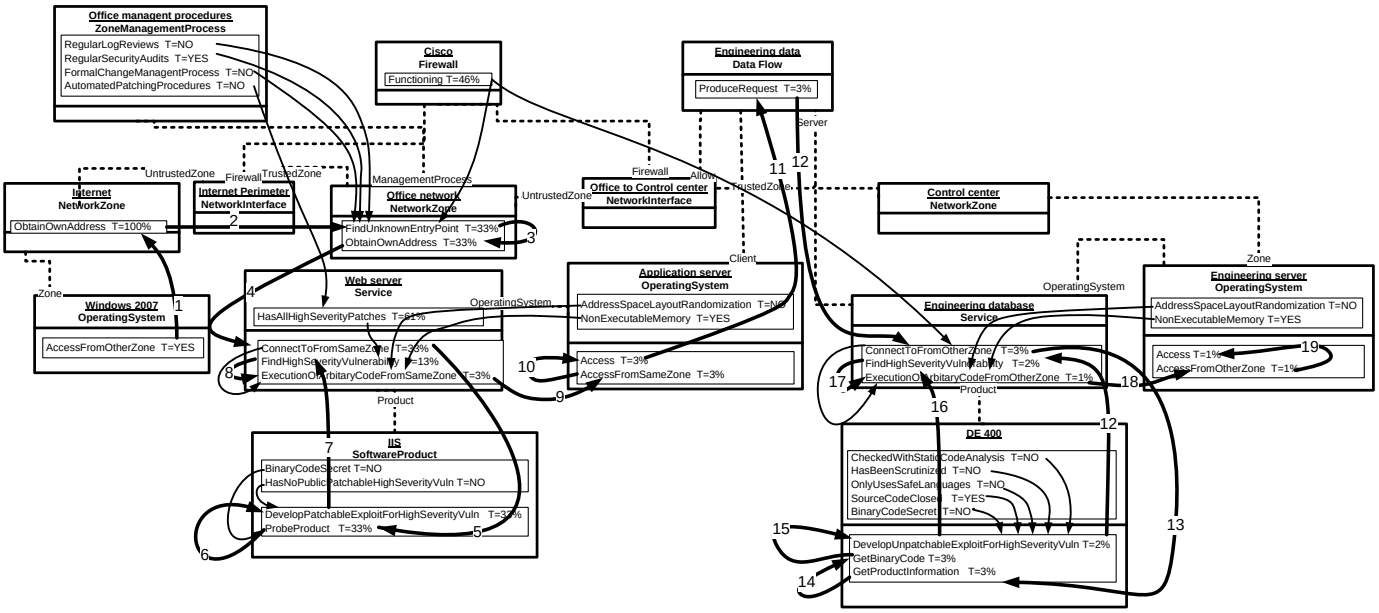

Fig. 2. An excerpt from an instance model of 19-step attack path together with the probabilities that each step along this path will be reached (T=True). The links in the attack path are the enumerated bold arrows.

VII. Summary and Future work

CySeMoL is a modeling language coupled to an inference engine for analyzing the security of enterprise system architectures. The inference engine produces attack paths from one attack step to another. For these attack paths, the inference engine estimates the probability that the attack can be accomplished by a professional penetration tester within one week using publicly available tools.

CySeMoL has been implemented in an existing tool [60] and validated on the component and system levels. On the component level, the theory specified in the dependencies is drawn from empirical studies in domain security and domain experts. On the system level, a Turing test suggests that the reasonableness of assessments produced by CySeMoL compares with that of a security expert and that both CySeMoL and the experts are more reasonable than security novices. These results suggest that CySeMoL would be useful where no security expert is available.

These results are promising. They suggest that assembling the body of system-security knowledge in a tool that can automate the assessments produced by experts in the field is feasible. Further work can be directed towards increasing CySeMoL's scope, refining and testing the model's accuracy, and maintaining and updating the theory.

When it comes to the scope, CySeMoL has been developed to support decision making related to the security of industrial control systems. This design focus has delimited the attacks that are covered by CySeMoL. Particularly, attacks on confidentiality are not well covered by CySeMoL because confidentiality is of lesser importance in industrial-control systems than in many other information systems. Further work is required if CySeMoL is to cover such attacks in a comprehensive manner. Effort can also be applied in modeling how attackers behave (i.e., determining which attacks attackers will attempt) and the consequences of successful attacks (i.e., to assess expected losses).

When it comes to accuracy, further tests are required to assess CySeMoL's accuracy with confidence. These tests can be on a component level and test a few probabilities or on system level and test the attack paths predicted for system architectures. Realization values can be sought in empirical tests, e.g., in conjunction with security tests or security exercises and competitions. Research can also be focused on refining the model and improving CySeMoL's accuracy. CySeMoL has been designed to produce assessments at a reasonable cost. In other words, it should not be overly costly to model a system-of-systems using CySeMoL. Work that refines the theory of CySeMoL by adding more detail to the metamodel to improve accuracy should take the cost of using these additions into consideration.

The threat environment and the countermeasures used at enterprises change over time. These changes will decrease CySeMoL's accuracy and value unless the theory is maintained and updated. As discussed in [65], some changes have a fundamental effect on the security domain. For example, when operating systems with containing new countermeasures become widely adopted. When fundamental changes occur, they are hopefully easy to identify along with the components of the theory they affect.

Other changes have limited impact on the overall threat environment or IT-landscape of enterprises. CySeMoL covers the most frequent of these changes. For instance, CySeMoL can detect the discovery of new vulnerabilities in a software

product. Smaller changes that are not covered by CySeMoL can be problematic to detect and adjust the theory for. Regular reviews of the theory (e.g., annual Turing tests) will be required to identify such gradual evolutions of the threat environment and the IT landscape. In any case, the theory will require ongoing study to preserve its accuracy.

REFERENCES

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

[2] B. Taskar et al., "Probabilistic Relational Models," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2007, pp. 129-175.

[3] V. Verendel, "Quantified security is a weak hypothesis: a critical survey of results and assumptions," *New Security Paradigms Workshop*, 2009.

[4] ISO/IEC, "Information technology -- Security techniques -- Information security management measurements, ISO/IEC 27004," Geneva, Switzerland, 2009.

[5] M. Swanson et al., "Security Metrics Guide for Information Technology Systems," National Institute of Standards and Technology, *NIST Special Publication* 800- 55, Gaithersburg, MD, USA, 2003.

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

[7] S. Bistarelli, F. Fioravanti., and P. Peretti., "Defense trees for economic evaluation of security investments," 2006, pp. 416-423.

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

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

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

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

[12] H. Pardue, J. Landry, and A. Yasinsac, "A risk assessment model for voting systems using threat trees and monte carlo simulation," in *Requirements Engineering for e-Voting Systems (RE-VOTE), 2009 First International Workshop on*, 2010, pp. 55–60.

[13] H. Pardue, J. P. Landry, and A. Yasinsac, "E-Voting Risk Assessment," *International Journal of Information Security and Privacy*, vol. 5, no. 3, pp. 19-35, 2011.

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

[15] M. McQueen et al.l, "Time-to-compromise model for cyber risk reduction estimation," *Quality of Protection*, 2006.

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

[17] T. Heberlein, et al. (2012, March 21). "A Taxonomy for Comparing Attack-Graph Approaches," [Online] *http://netsq.com/Documents/AttackGraphPaper.pdf*.

[18] S. Roschke et al., "Towards Unifying Vulnerability Information for Attack Graph Construction," in *Proceedings of the 12th International Conference on Information Security*, 2009, p. 233.

[19] L. P. Swiler et al., "Computer-attack graph generation tool," in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, 2000, pp. 307-321.

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

[21] R. Lippmann, "Netspa: A network security planning architecture," M.Eng. thesis, Massachusetts Institute of Technology, 2002.

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

[23] J. Homer, et al., "A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks," Kansas, 2010.

[24] S. Noel et al., *Advances in Topological Vulnerability Analysis*. Washington, DC: IEEE, 2009, pp. 124-129.

[25] R. P. Lippmann and L. L. C. Williams, "GARNET: a Graphical Attack graph and Reachability Network Evaluation Tool," in *Visualization for Computer Security*, K. Prole, Ed. Heidelberg-Berlin: Springer Berlin / Heidelberg, 2008, pp. 44-59.

[26] M. Chu et al., "Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR," in *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, 2010, pp. 22–33.

[27] R. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *13th European Symposium on Research in Computer Security (ESORICS)*, 2008, no. 716665, pp. 18-34.

[28] H. Holm et al., "A quantitative evaluation of vulnerability scanning," *Information Management & Computer Security*, vol. 19, no. 4, 2011.

[29] K. Stouffer, J. Falco, and K. Kent, "Guide to Industrial Control Systems (ICS) Security Recommendations of the National Institute of Standards and Technology," *Nist Special Publication*, vol. 800, no. 82, 2008.

[30] K. Ingols et al., "Modeling Modern Network Attacks and Countermeasures Using Attack Graphs," in *Annual Computer Security Applications Conference*, 2009, pp. 117-126.

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

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

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

[34] The MITRE Corporation (2012, March 21), "The Common Attack Pattern Enumeration and Classification," *(website)*, 2011. [Online]. Available: http://capec.mitre.org/.

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

[36] C. Cowan et al., "Buffer Overflows : Attacks and Defenses for the Vulnerability of the Decade," in *Foundations of Intrusion Tolerant Systems, 2003 [Organically Assured and Survivable Information Systems]*, 2003, pp. 227-237.

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

[38] I. Simon, (2012, March 21) "A comparative analysis of methods of defense against buffer overflow attacks," 2001, [Report]. Available: *http://www. mcs. csuhayward. edu/\~ simon/security/boflo. html*

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

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

[41] M. Dell' Amico et al., "Password Strength: An Empirical Analysis," *2010 Proceedings IEEE INFOCOM*, pp. 1-9, 2010.

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

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

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

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

[46] T. Sommestad, H. Holm, and M. Ekstedt, "Effort estimates for vulnerability discovery projects," in *HICSS'12: Proceedings of the 45th Hawaii International Conference on System Sciences*, 2012.

[47] T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of remote arbitrary code execution attacks," *Information Management & Computer Security*, (Accepted) .

[48] T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of Denial-of-Service attacks," in *TrustCom 2011*, 2011, no. 1.

[49] T. Sommestad et al., "Quantifying the effectiveness of intrusion detection systems in operation through domain experts," (submitted).

[50] R. Cooke, “TU Delft expert judgment data base,” *Reliability Engineering & System Safety*, vol. 93, no. 5, pp. 657-674, May 2008.
[51] T. Sommestad, “Exploiting network configuration mistakes: practitioners self-assessed success rate,” KTH, Technical Report TRITA-EE 2011:069, Stockholm, Sweden, 2011.
[52] T. Sommestad et al., “Security mistakes in information system deployment projects,” *Information Management and Computer Security*, vol. 19, no. 2, 2011.
[53] A. Wool, “A quantitative study of firewall configuration errors,” *Computer*, pp. 62–67, 2004.
[54] J. R. Jacobs, “Measuring the Effectiveness of the USB Flash Drive as a Vector for Social Engineering Attacks on Commercial and Residential Computer Systems,” Embry Riddle Aeronautical University, 2011.
[55] S. Stasiukonis, “Social engineering, the USB way,” *Dark Reading*, vol. 7, 2006.
[56] T. N. Jagatic et al., “Social phishing,” *Communications of the ACM*, vol. 50, no. 10, pp. 94–100, Mar. 2007.
[57] R. Dodge and A. Ferguson, “Using Phishing for User Email Security Awareness,” in *Security and Privacy in Dynamic Environments*, vol. 201, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds. Springer Boston, 2006, pp. 454-459.
[58] M. Buschle (2012. March 21), “KTH | The Enterprise Architecture Tool,” 2011. [Online]. Available: http://www.kth.se/ees/omskolan/organisation/avdelningar/ics/research/eat.
[59] R. M. O’Keefe and D. E. O’Leary, “Expert system verification and validation: a survey and tutorial,” *Artificial Intelligence Review*, vol. 7, no. 1, pp. 3-42, Feb. 1993.
[60] M. Buschle et al., “A Tool for Enterprise Architecture Analysis using the PRM formalism,” in *Proc. CAiSE Forum 2010*, 2010.
[61] M. J. Druzdzel, “GeNIe: A development environment for graphical decision-analytic models,” in *Proceedings of the 1999 Annual Symposium of the American Medical Informatics Association*, 1999, p. 1206.
[62] R. Agarwal, R. Kannan, and M. Tanniru, “Formal validation of a knowledge-based system using a variation of the Turing test,” *Expert Systems with Applications*, vol. 6, no. 2, pp. 181-192, Apr. 1993.
[63] M. Österlind, “Validering av vektyget Enterprise Architecture Tool,” Royal Institute of Technology (KTH), 2011.
[64] M. Buschle et al., “A Tool for automatic Enterprise Architecture modeling,” in *CAISE’11 Forum*, 2011.