Security Optimization of Dynamic Networks with Probabilistic Graph Modeling and Linear Programming

Hussain M.J. Almohri, *Member, IEEE*, Layne T. Watson, Danfeng (Daphne) Yao, *Member, IEEE* and Xinming Ou

Abstract—
Securing the networks of large organizations is technically challenging due to the complex configurations and constraints. Managing these networks require rigorous and comprehensive analysis tools. A network administrator needs to identify vulnerable configurations, as well as tools for hardening the networks. Such networks usually have dynamic and fluidic structures, thus one may have incomplete information about the connectivity and availability of hosts. We describe a probabilistic graph model and several algorithms for analyzing and improving the security of large networks. We demonstrate their use in solving several types of useful network security management problems. Among them is the optimal placement problem, where the network administrator needs to compute on which machine(s) to install new security products in order to maximize the security benefit for the organizational network. In comparison to related solutions on attack graphs, our probabilistic model provides mechanisms for expressing uncertainties in network configurations, which is not reported elsewhere. Our computation utilizes advanced sequential linear optimization techniques and is scalable to large networks. We have performed comprehensive experimental validation with real-world network configuration data of a sizable organization.

1 Introduction

Large organizations need rigorous security tools for analyzing potential vulnerabilities in their networks. However, managing large-scale networks with complex configurations is technically challenging. For example, organizational networks are usually dynamic with frequent configuration changes. These changes may include changes in the availability and connectivity of hosts and other devices, and services added to or removed from the network.

Network administrators also need to respond to newly discovered vulnerabilities by applying patches and modifications to the network configuration and security policies, or utilizing defensive security resources to minimize the risk from external attacks. For instance, to prevent a remote attack targeting a host it is useful to analyze the candidate defensive strategies in choosing installation and runtime parameters for one or several intrusion prevention system.

To facilitate a scalable security analysis of organizational networks, attack graphs (e.g., [1], [2]) were proposed. Attack graphs show possible attack paths with respect to a particular network setting, which provide the necessary elements for modeling and improving the security of the network.

Existing work utilizes attack graphs (for example, [1], [2], [3]) for analyzing the security risks by quantifying attack graphs using a variety of techniques such as Bayesian belief propagation [4], [5], [6], [7], basic laws of probability [8], [9], and vertex ranking algorithms [10], [11]. These models lack a systematic and scalable computation of optimized network configurations. Current attack graph quantification models assume a network with known and fixed configurations in terms of the connectivity, availability and policies of the network services and components disregarding the dynamic nature of modern networks. Moreover, except a few attempts [12], [13], [14], [6], previous work has solely focused on computing a numerical representation of the risk without addressing the more challenging problem of risk management and reduction.

In this paper, we present a rigorous probabilistic model that measures the security risk as the probability of success in an attack. Our probabilistic model referred to as the *success measurement model* has three main features: *(i)* rigorous and scalable model with a clear probabilistic semantic, *(ii)* computation of risk probabilities with the goal of finding the maximum

• *H. M. J. Almohri is with the Department of Computer Science, Kuwait University, Kuwait. Email: almohri@cs.ku.edu.kw.*

• *L. T. Watson is with the Departments of Computer Science and Mathematics, Virginia Tech, Blacksburg, VA 24060. Email: ltw@cs.vt.edu.*

• *D. Yao is with the Department of Computer Science, Virginia Tech, Blacksburg, VA 24060. Email: danfeng@cs.vt.edu.*

• *X. Ou is with the Department of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506. Email: xou@ksu.edu.*

attack capabilities, and *(iii)* considering dynamic network features and the availability of mobile devices in the network.

As an application of our success measurement model, we formalize the problem of utilizing network security resources as an optimization problem with the goal of computing an *optimal placement* of security products across a network. Our new contribution is to define this optimization problem and provide an efficient algorithm based on a state of the art technique called sequential linear programming. Our algorithm is proved to converge and it is scalable to large networks with thousands of components an attack paths. Our contributions in this paper include:

- *A scalable probabilistic model* that uses a Bernoulli model to measure the risk in terms of the probability of success to achieve an attack goal.
- *Efficient security optimization model* that is generated based on a quantified attack graphs and computes an optimal placement of security products according to organizational and technical constraints.
- *Modeling dynamic network features* for a realistic and accurate analysis of the risk associated with modern networks.

The results of our experiments confirm three key properties of our model. First, the vulnerability values computed from our model are accurate. Our manual inspection of the results confirm that the ECSA values obtained in the experiments correlate to the vulnerabilities of components in the network. Second, our security improvement method efficiently finds the optimal placement of security products subject to constraints. Third, we quantify the additional vulnerabilities introduced by mobile devices of a dynamic network. Our results indicate that an infected mobile device within the trusted region creates a preferred attack direction towards the attack target, which increases the chance of success at the target host.

2 Related Work

Probabilistic metrics. Using the probability theory to compute a quantitative security has been recently reported [4], [5], [9]. A work by Wang et al. [9] considers a probabilistic model for computing a security risk metric using attack graphs. The work in [9] discusses an interpretation of the metric and a heuristic to compute the metric. Our success measurement model generalizes this work by capturing the uncertainty in attacker's choices (discussed as a random selector in Section 4).

Bayesian analysis of attack graphs [6], [5], [7] differs with our success measurement model in that our model does not require the knowledge of conditional probabilities. In [5], a dynamic Bayesian network model was proposed that is capable of incorporating temporal factors. Xie et al. [7] introduced a Bayesian model that adds a node to the Bayesian network indicating whether or not an attack has happened. Although this extension improves the models in [5], it does not capture the various possibilities of attack paths taken by an attacker before reaching an intermediate attack goal.

None of the previous work considers the effect of device availability on open networks. Furthermore, optimized network configurations and improvement in our work has not been previously studied. Bayesian methods are powerful in computing unobserved facts, such as predicting possible threats. It remains unclear how Bayesian methods can be used to support variability in attacker's decisions, device availabilities, and the effect of mobile devices.

Attack graph ranking. PageRank is an algorithm proposed by Page et al. [21], which is used to rank important web pages. The idea of page rank is based on a random web surfer that follows the links on web pages and compute a priority rank. Due to the similarity between link graphs and attack graphs, a variety of successful research has proposed the use of a modified version of PageRank to rank attack graphs. A ranking algorithm based on PageRank [21], AssetRank [11] was proposed to rank any dependency attack graph using a random walk model. AssetRank is a generalization of PageRank extending it to handle both conjunctive and disjunctive nodes. AssetRank is supported by an underlying probabilistic interpretation based on a random walk. Mehta *et al.* propose a ranking method using state enumeration attack graphs [10]. The idea of PageRank is applied to state enumeration attack graphs with a modified interpretation of the ranking. Attack graphs based on model checking have been proposed in [16] formalizing an intrusion attack in a finite state model. Authors in [16] do not propose a complete attack graph ranking method. Instead, a method to compute minimal critical attack assets based on user-specified metrics has been introduced.

Other approaches to security assessments include a goal-motivated attacker model based on a Markov decision process [22], a weakest-adversary approach to ranking attack graphs [23], a generic framework for an attack resistance metric [24], and an enterprise IT risk metric using CVSS scores [25].

The aforementioned techniques do not consider the effect of device availability in their vulnerability ranking algorithms. In addition, recommendation of security hardening options is not addressed. While we provide a systematic way to find optimal recommendation options, other researches have not provided such a mechanism.

Security hardening. The authors in [6], [13] formulated the optimal security hardening problem as a multi-objective optimization problem. In [14] Noel and Jajodia presented a greedy algorithm to solve the problem of the best placement of IDS sensors in a network using attack graphs. It is to find a minimal number of sensors that can cover all critical attack paths. Also, [26] describes a method on finding the initial conditions that need to be removed to improve the network security. Our approach differs in that we use mathematical programming to formulate a new problem to compute the best placement of a set of security improvement options on a subset of rule nodes of an attack graph. In contrast to a genetic algorithm used in [13], we use a state of the art technique, named sequential linear programming, which is scalable and efficient [17], [18], [27], [28].

The work in [12] defines a cost function to measure the effect of various network reconfigurations. The proposed method follows a forward search approach to assess the result of network hardening options.

Huang et al. proposed a method for distilling critical attack graph surface iteratively through minimum-cost SAT solving [29]. The presented method is useful in finding the most critical attack path, which can be considered later for hardening the security of the network. Such a result can be used to guide our improvement recommendation method to consider hosts found on a critical path.

In [8] the authors provided a method to quantify the attack graph and simulate attackers' choices to compute an improved reconfiguration. While being a valuable approach, the proposed method does not take into account the the availability of machines and uncertainty in attackers' decisions.

The authors in [8], [14], [26], [29] propose methods for hardening the security of networks. However, the recommendation of security improvement options is not studied there.

ADEPS [30] uses IDS alerts from some nodes on an attack graph to calculate the likelihood that an attack can occur on other nodes. It then computes the response to the likely attacks based on a repository of responses. Our computation differs from ADEPS in that ours has a rigorous mathematical foundation, which allows us to solve more complex security optimization problems.

Our work differs from the existing work presented above. First, we provide a general mechanism for capturing network component availability (i.e., the variability in a device's network reachability), which also leads to quantifying and analyzing possible threats from mobile devices such as laptops, tablet computers, and cellphones. Second, our probability calculation scheme is general enough to allow performing various levels of success probability analysis by introducing variable attack steps as part of success probability computation. Third, we complete the analysis of network security threats by providing a sound and computationally efficient security improvement recommendation technique that is capable of finding optimal network configurations as well as optimal placement of security solutions in the network.

3 Overview

Our attack-graph work is motivated by the *optimal placement* problem. Intuitively, it is the problem of how to minimize the security risk for the ultimate attack goal through the optimal placement of security products. Specifically, the optimal placement problem is to compute the most effective placement of T number of security products with K_τ placement options for each improvement option τ. The optimal placement may be subject to certain placement constraints, such as limiting the deployment of security products to a subset of machines in the network.

To solve the optimal placement problem, one needs to quantitatively compare all the possible configurations or placement options. A bruteforce search approach for solving this combinatorial problem has a worst-case factorial complexity in the number of placements, which is clearly impractical. With attack graphs, we can efficiently measure the effectiveness of a security product based on the percentage reduction in the chance of a successful attack on a particular target in the network.

In general, there are several technical problems for hardening the security of a network: *i) modeling*, i.e., how to express and model the placement options in attack graph, *ii) quantification*, i.e., a realistic representation of the security risk in a network with dynamic properties, and *iii) computing*, i.e., how to efficiently compare these different configurations.

For computing vulnerabilities in the network, we design a probabilistic *success measurement* model. The model quantifies the vulnerabilities of networked components and resources, by computing the expected chances of successful attacks (ECSA) on attack graph nodes.

We define *the ECSA of a node u* as the expected chance that a attacker successfully exploits the node u of the attack graph, given certain initial belief. There may be multiple attack paths (representing multiple attack choices) to reach u. Thus, the ECSA for u must represent an estimation of the success as a combined value of success for previous nodes.

What differs our work from existing attack graph modeling is two folds.

1. Our model allows one to represent uncertain network properties and incorporates them in the vulnerability definitions.
2. Our computation methods reveal the vulnerabilities in the network and quantitatively correlate them with dynamic network properties.

In our success measurement model, the basic computation requires two types of inputs, and outputs the expected chances of successful attacks against network components and resources (Figure 2). One type of the input is an attack graph generated by a attack graph generator tool. The other input is a set of initial belief values associated with the ground facts

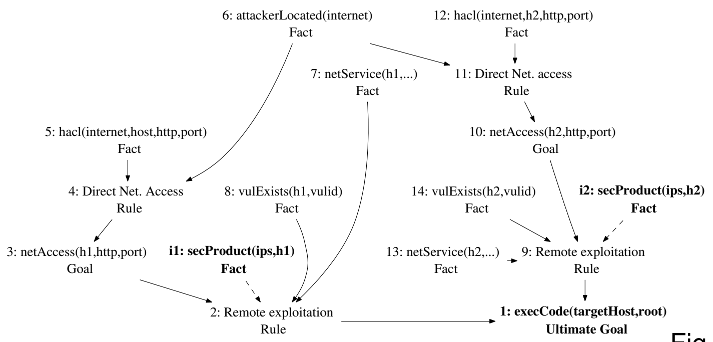

Fig. 1. An attack graph with multiple candidate placements (either host1 or host2) of an intrusion prevention system. The problem is to minimize the probability of a successful in achieving the ultimate attack goal by choosing one placement between i_1 and i_2.

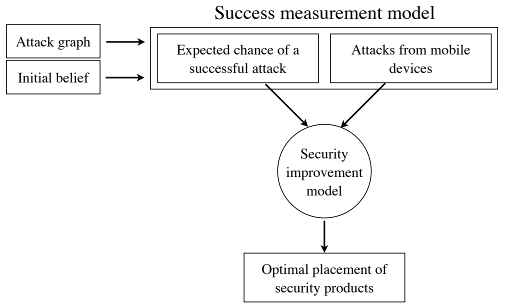
Success measurement model
Attack graph
Initial belief
Expected chance of a successful attack
Attacks from mobile devices
Security improvement model
Optimal placement of security products

Fig. 2. Given an attack graph and a minimal set of initial belief values associated with fact nodes, success measurement model computes the expected chance of a successful attack with a consideration of dynamic availability of mobile devices. The computed expected values are used in the improvement model to find an optimal placement of security products.

which correspond to fact nodes of the attack graph. Facts related to network configurations and vulnerability data are associated with success probabilities and constitute our initial belief. Our model requires a minimal set of initial belief values (Section 6) that can be assigned according to an estimation obtained from experts' knowledge and standard vulnerability scoring systems such as the common vulnerability assessment system (CVSS) [15].

Our success measurement model forms the basis for solving the problem of computing an optimal placement. Figure 1 shows two improvement Nodes i_1 and i_2 (defined in Section 5) added to the attack graph. Fact Node i_1 corresponds to the placement of an intrusion prevention system (IPS) at Node 2, and fact Node i_2 corresponds to the placement of an IPS at Node 9. Each of the placements can make the target less vulnerable. The optimal placement is to find the placement of one or more improvements (i.e., security products) that best lowers the vulnerability of the target. We provide the model, formulation and computation for efficiently solving the problem for large attack graphs.

Features and Capabilities. Our approach demonstrates a unique mechanism for optimizing the security configuration in a large network. A list of features follows.

- *Scalable modeling.* Attack stages and paths are expressed in a mathematical programming model. The benefit of this approach is that a well-known mathematical programming model can be solved using existing state of the art algorithms. Further, various complex deployment and configuration constraints can be expressed using existing techniques. Examples of such constraints include but is not limited to (i) choosing a subset of available security products, (ii) constraints on installation of products on specific hosts, (iii) constraints on co-installation of a subset of products (such as if product t_1 is installed on host h_1, then t_2 may not be installed on h_1), and (iv) performance constraints by assigning performance parameters to the model.
- *Computing maximal attack capabilities.* A feature of our approach is the computation of risk measurements in a maximization problem. This approach allows to compute the highest points of risk in the network.
- *Attack uncertainty.* We present a modeling mechanism to analyze the uncertainty in attack choices that are available when attackers could take alternative paths. We numerically analyze this uncertainty as part of our probabilistic analysis of security risk.

4 Success Measurement Model

In this section we present our success measurement model to compute the expected chance of a successful attack on a network with respect to the attack's ultimate goal. We first present the definitions of the expected chance of a successful attack (ECSA) followed by the description of two efficient algorithms to compute ECSA values.

4.1 Definitions of ECSA Values

The key component of our success measurement model is the probabilistic definition of the expected chance of a successful attack against any node in the attack graph.

We present an alternative approach to the Bayesian analysis discussed in [6], [7]. Our success measurement model computes probabilities as a function of initial belief probabilities without the need for specifying conditional probabilities required by Bayes' theorem. The set of initial belief values required by our model is small and can be obtained from standard vulnerability assessment systems (discussed in Section 6).

Our model measures the success of an attacker based on the attack dependencies determined by a logical attack graph.

Definition 1: A logical attack graph $G = (V, E)$ is a digraph where $V = N_f \cup N_g \cup N_r$ and N_f, N_g, N_r are disjoint sets of nodes containing fact nodes, goal nodes, and rule nodes, respectively. E is the set of arcs, and $\mathcal{G} \in N_g$ is the attacker's goal.

We define the sample space for a node and a corresponding random variable representing attack outcomes. The outcome of an attack attempt on a node can either by a success or a failure. Let $\Omega(u)$ be the sample space for a node $u \in V$ for an attack graph G. We define the random variable X_u for the node u as a Bernoulli random variable with $X_u(\omega) = 1$ denoting success in an attack and $X_u(\omega) = 0$ failure, where ω is an outcome.

Definition 2: For any node $u \in V$ of an attack graph, the expected chance of a successful attack (ECSA) at a node u is given as $E[X_u] = P(X_u = 1)$, that is, the probability of success for the random variable X_u.

Let $\phi(u) = \{v \mid (v, u) \in E\}$ be the set of predecessors (dependencies) of a node u. In the following, we define ECSA for the derived nodes based on the corresponding logical semantics (that is, conjunction for a rule node and disjunction for a goal node).

ECSA value of a rule node. Let $u \in N_r$ be a rule node and $\phi(u) = \{v_1, v_2, \dots, v_t\}$. The random variable X_u — corresponding to the success or failure of the attacker at node u — is defined as the product of the random variables for all predecessor nodes $v \in \phi(u)$, for which the expected value is

$$E[X_u] = \prod_{v \in \phi(u)} E[X_v], \tag{1}$$

assuming independence of the predecessor random variables (further discussed in Section 4.3).

ECSA value of a goal node. An attack graph has several goal nodes. A goal node either depends on a single exploitation rule (represented by a rule node) or multiple exploitation rules such as u_1 in Figure 3.

A goal node with multiple rule node dependencies is a logical disjunction. In reality, this disjunction indicates that there are multiple attack choices for an attacker towards a specific attack goal. For instance, consider a server with a local privilege escalation vulnerability (which is exploitable remotely in a multi-step attack) and runs a network service with multiple remote vulnerabilities. An attacker must exploit one (or more) of these vulnerabilities to gain privileges on the target server. In the lack of observable evidence, one needs to compute the ECSA of a goal node with a function that correctly captures the probabilities of such attack choices.

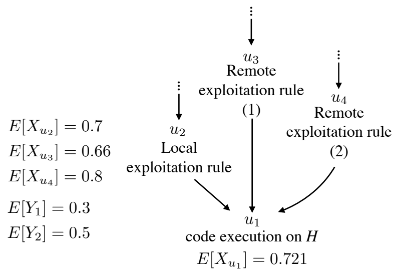

Fig. 3. A goal node for an attack on host H with three attack choices: a local exploitation and two methods of remote exploitation. The variables Y_1 and Y_2 measure the probability of attack choices. We assume $E[Y_1]$ and $E[Y_2]$ are not available, and thus, we computationally determine their values based on Equation 2.

Our approach is to computationally determine attack choice probabilities according to various attack patterns (Section 4.2). Per our knowledge, no previous work has modeled this reality.

In the the attack graph of Figure 3, Node u_1 has three predecessors (rule Nodes u_2, u_3, and u_4). To compute $E[X_{u_1}]$, we introduce auxiliary Bernoulli random variables Y_i (referred to as the random selectors) to capture the random selection of an attack path. The values of Y_i are multiplied with the computed ECSA for the predecessor nodes to reflect the attack choices. In Section 4.2, we show how the values of Y_i variables are computed.

Let $\phi(u) = \{v_1, v_2, \dots, v_t\}$ be the set of dependencies of u. Then we define the random variable X_u for a goal node $u \in N_g$ for which the expected value is determined as

$$E[X_u] = \sum_{k=1}^{t-1} \left[E[Y_k] E[X_{v_k}] \prod_{i=1}^{k-1} (1 - E[Y_i]) \right] + E[X_{v_t}] \prod_{i=1}^{t-1} (1 - E[Y_i]). \tag{2}$$

Observe that the Definition (2) selects $X_u = X_{v_i}$ by the event $Y_i = 1$, $Y_j = 0$ for $j < i < t$ (for example, Figure 3). Note that the Bernoulli variables Y_i in general depend on the node u, but this dependence is not reflected with the notation $Y_i^{(u)}$ for simplicity.

While we generally assume that the random variables are independent, we do not assume they are mutually exclusive. That means multiple concurrent attacks (more than one Y_i) can lead to a goal node from various attack paths.

Our success measurement model is versatile and powerful because of its capability to express and analyze options and uncertain choices. It can be easily modified to analyze many network scenarios, which we demonstrate in Section 5. For example, for solving the optimal placement problem, one may augment the model with a special type of fact node representing the installation of a security product. We discuss this extension in detail in Section 5.

4.2 Computing ECSA Values

Attack choices are uncertain, and various attack scenarios are possible. Existing work such as [16], [9], [11], have provided ways to compute a static view of the security risk corresponding to specific attack scenarios. In this section we describe the method for computing ECSA values of an attack graph with a goal of finding the highest possible chances of success for an attack.

4.2.1 The most vulnerable components

The computation method described in this section allows one to find the ECSA values such that the ECSA of the attack target is maximized. The result of this computation is in particular important for optimal placement of security hardening products described in Section 5.1.

To find the most vulnerable components, we formulate a maximization problem with a nonlinear objective function subject to linear and nonlinear equality constraints. The decision variables are the nodes of an attack graph. The equations for computing ECSA, (1) and (2), form the constraints of the maximization problem.

Let x_i be a decision variable for a node $i \in N_r \cup N_g$ corresponding to $E[X_i]$, and $x = (x_1, x_2, \dots, x_M)^T$ be the vector of unknown ECSA values for all nodes. Let y_i be a decision variable for a random selector Y_i corresponding to $E[Y_i]$, and $y = (y_1, y_2, \dots, y_P)^T$ be the vector of unknown expected values of the random selectors. For a rule node $u \in N_r$ with predecessors $\phi(u)$, the constraint function is

$$f_u(x,y) = \begin{cases} x_u - x_j \prod\limits_{\substack{k\in\phi(u)\\ k\in N_f}} P(X_k = 1), & j \in \phi(u) \cap N_g, \\ x_u - \prod\limits_{\substack{k\in\phi(u)\\ k\in N_f}} P(X_k = 1), & \phi(u) \cap N_g = \emptyset. \end{cases} \tag{3}$$

Note that Equation 3 has two cases. The first case is for rule nodes with one goal node as a predecessor and the second case is for rule nodes with no goal nodes as predecessors. For a goal node $u \in N_g$ with predecessors $\phi(u) = \{v_1, v_2, \dots, v_t\}$, the constraint function is

$$f_u(x,y) = x_u - \sum_{k=1}^{t-1} \left[y_{m_u+k} x_{v_k} \prod_{i=1}^{k-1} (1 - y_{m_u+i}) \right] - x_{v_t} \prod_{i=1}^{t-1} (1 - y_{m_u+i}). \tag{4}$$

All the selector variables for all the goal nodes are numbered consecutively, so that the y_i for node u are $y_{m+1}, y_{m+2}, \dots, y_{m+t-1}$ for some $m = m_u$ depending on u.

Let $f(x,y) = (f_1, f_2, \dots, f_M)^T$ be a vector-valued function. The nonlinear program for finding the most vulnerable components is

$$\begin{aligned} &\text{maximize } f_{\mathcal{G}}(x,y) \\ &\text{subject to } f(x,y) = 0, \\ &0 \le x_i \le 1 \text{ , } i = 1, \dots, M, \\ &0 \le y_i \le 1 \text{ , } i = 1, \dots, P. \end{aligned} \tag{5}$$

In (5), the vector-valued function $f(x,y)$ holds all the constraint functions (that is, (3) and (4)) for all rule and goal nodes in the attack graph. Note that the constraints in $f(x,y)$ are the ECSA equations (1) and (2) equalized to zero.

To solve (5), we use a technique called sequential linear programming (SLP) [17]. SLP has been widely applied in engineering, and efficient algorithms for solving nonlinear programs using SLP are known. SLP is a standard technique for finding a close approximate solution for nonlinear optimization problems. SLP is computationally efficient and converges to an optimal solution [18].

4.3 Attack Dependencies

A major problem in probabilistic risk assessments is to accurately capture attack steps dependencies and correlations. Attack dependencies in the form of attack preconditions are intrinsically captured by our model. That is because we base our analysis on attack graphs that are formed based on the dependency relations among the nodes. Therefore, the probabilities of success are fundamentally propagated using the dependency relations determined in an attack graph. Another form of attack dependencies is *attack step dependencies*. In this section, we define attack step dependency and show that with our current analysis assuming independence is reasonable.

Definition 3: An attack step represented by a goal or rule node u in an attack graph is dependent on another attack step v, if achieving v affects the decision of the attacker in achieving u.

The dependency as defined in Definition 3, occurs when a dependent node u is a direct or indirect successor of v. The only way u can be dependent on v is if v is known to have $X_v = 1$. Knowing $X_v = 1$ means an attack has succeeded, and the attacker is now using that knowledge to stage a second attack. In our current model, we assume independence of all attack steps since the scope of this paper is limited to analyzing a single series of attacks. The attack step dependencies could occur when multiple consequent attacks are analyzed. Thus, we leave the study of attack step dependencies for a future work.

5 Applications

Using the rigorous probabilistic model introduced in Section 4.1, we define and solve an optimal placement problem and extend our analysis to include the effect of mobile devices. These two applications of our model are summarized as follows.

- *Optimal placement.* Given a set of security hardening products (e.g., a host based firewall), we compute an optimal distribution of these resources subject to placement constraints.
- *Machine availability and the effect of mobile devices.* Our work is the first to show how to represent and assess devices with variable availability (frequently joining and leaving the network), which is one of the characteristics of mobile devices with variable connectivity.

Our success measurement model and its computational techniques naturally yield the solutions to these problems, which are described below. In Section 7, experiments to evaluate the applications of our model are presented.

5.1 Optimal Placement of Security Products

With limited resources for hardening an organizational network, it is important to install a single or a combination of security hardening products so that the expected chance of a successful attack on the network is minimized. To find the best placement of a set of security products in a network, we extend the attack graph to define a security product as a special fact node referred to as an *improvement node*, which is a fact node that represents a security hardening product, service, practice, or policy.

The objective of solving the problem of optimal placement of security products is *to compute and compare the effects of various placements of one or more improvement nodes while subject to certain constraints, and choose the placement that minimizes the attack goal's ECSA value.*

The following describes computing the best place to deploy a single security product (that can be generalized to multiple security products) in the network. We formulate this optimal placement problem as a minimax problem — finding the best placement of the improvement option that minimizes $\hat{x}_{\mathcal{G}}$, where $\hat{x}_{\mathcal{G}}$ is the maximum of $E[X_{\mathcal{G}}]$ with respect to X_u and $Y_i^{(u)}$.

We consider a single improvement option for rule nodes given deployment constraints. We define the set of admissible rule nodes $N_{ra} \subseteq N_r$ as a subset of all rule nodes. Let $P(X_\tau = 1)$ be the initial belief of some improvement option τ. The problem is to find a configuration that minimizes $\hat{x}_{\mathcal{G}}$. That is, we aim to find a rule node $u \in N_{ra}$ such that if $\tau \in \phi(u)$, the value of $\hat{x}_{\mathcal{G}}$ is minimized.

Let $\mathcal{A} = |N_{ra}|$ and $j_1 < j_2 < \ldots < j_{\mathcal{A}}$ be the nodes in N_{ra}. Define 0-1 variables t_{j_i} for $i = 1, \ldots, \mathcal{A}$ and let $T = (t_{j_1}, \ldots, t_{j_{\mathcal{A}}})$. A single improvement corresponds to the constraint

$$t_{j_1} + t_{j_2} + \cdots + t_{j_{\mathcal{A}}} = 1,$$

and the generalization to multiple improvements is obvious.

We modify the definition of $f_u(x, y)$ for a rule node given in Equation (3) to include the effect of the improvement option τ. For a rule node $u \in N_{ra}$, define

$$f_u(T, x, y) = \tag{6}$$
$$\begin{cases} x_u - (P(X_\tau = 1))^{t_u} x_j \prod\limits_{\substack{k \in \phi(u) \\ k \in N_f}} P(X_k = 1), & j \in \phi(u) \cap N_g, \\ x_u - (P(X_\tau = 1))^{t_u} \prod\limits_{\substack{k \in \phi(u) \\ k \in N_f}} P(X_k = 1), & \phi(u) \cap N_g = \emptyset. \end{cases}$$

This modified definition adds the improvement node at exactly one rule node in N_{ra}. Note that the definition of f_u for a goal node is identical to Equation 4. The minimax problem to find the best placement of security products is

$$\begin{aligned} &\underset{T \in \{0,1\}^{\mathcal{A}}}{\text{minimize}}\ \hat{x}_{\mathcal{G}} \\ &\text{subject to } t_{j_1} + \cdots + t_{j_{\mathcal{A}}} = 1, \end{aligned} \tag{7}$$

where $\hat{x}_{\mathcal{G}}$ is the solution to

$$\begin{aligned} &\underset{x,y}{\text{maximize}}\ f_{\mathcal{G}}(T, x, y) \\ &\text{subject to } f(T, x, y) = 0, \\ &0 \le x_i \le 1, i = 1, ..., M, \\ &0 \le y_i \le 1, i = 1, ..., P. \end{aligned} \tag{8}$$

The minimax problem (7) maximizes the ECSA value of the attack's goal ($E[X_{\mathcal{G}}]$) to find the highest chance of success in attacking a specific network component (such as a server). The result of the inner maximization problem (8) is then used in the outer minimization problem (7) to find the best placement of the security product such that the maximized ECSA is minimized.

The inner maximization problem is solved using SLP as before. The outer minimization problem is a limited combinatorial problem for one improvement. For multiple improvements, the outer problem can be solved by an LP relaxation (change $t_i \in \{0, 1\}$ to $0 \le t_i \le 1$) with branch and bound. For k improvements, the complexity is $\binom{\mathcal{A}}{k}$.

5.2 Threat from Mobile Devices

To capture the increase of security threats due to the inclusion of mobile devices (such as laptops, smartphones, and tablet computers) in the network, our approach is to extend an original attack graph for a network to include attack paths from mobile devices. Specifically, we define special rules to represent the uncertain availability of mobile devices in an attack graph, as well as the corresponding ECSA formulation and computation. The ability to model the availability of machines in attack graphs is general and useful beyond the specific mobile devices studied.

Attack graph extension. We extend the rules of the MulVAL attack graph generator [19] to include exploitation rules that capture the availability of mobile devices. An identified mobile device may not always appear in the network. Mobile devices rarely include a server software. The majority of Internet-based mobile applications are clients to the outside world, requiring interaction with malicious input to execute a successful exploit. For instance, most of the vulnerabilities that we studied for the Android platform involved an interaction with a malicious code (i.e. a malicious website) and exploiting a local vulnerability. Thus, we define basic exploitation rules for mobile devices in Figure 4.

```
execCode(H,Perm) :-
        compromised(H),
        vulExists(H,Vulid,
                  localExploit,privEscalation).
compromised(H) :-
        deviceOnline(H,Platform),
        vulExists(H,Vulid,remoteClient,
                      codeExecution),
        maliciousInteraction(H,_,App).
```

Fig. 4. The two predicates describe attack stages (i.e., remote and local exploits). The predicate deviceOnline(H,Platform) captures the availability of the device H.

We capture the availability of a device with the node `deviceOnline(H,Platform)`. In the success measurement model, these nodes are dynamic nodes with no fixed initial belief. The availability of a device may be measured as the percentage of the time that the device is connected within the target network (e.g., through a wireless connection) in a certain period. This data may be collected or estimated for the target network.

Our rules are general enough to be applicable to any logical attack graph generator. In addition to the extended rules, an input of mobile devices data is given to the attack graph generator. The data includes access information to other devices, presence of vulnerabilities, and information about the platform. Once the attack graph is generated, we compute the ECSA values for the network, which is described next.

ECSA for mobile devices. For mobile device fact nodes, the availability of the device cannot be deterministically specified. Thus, fact nodes similar to `deviceOnline(H,Platform)` cannot have a precomputed value for all instances of ECSA computation. A fixed value of $E[X_u]$ (for a fact node u) does not accurately reflect the device's availability. In order to solve this issue, we define a *stochastic fact node* as a fact node that represents a dynamic ground fact that is not associated with a fixed initial belief.

We define a random variable for a stochastic fact node (such as `deviceOnline(H,Platform)`). Based on our success measurement model, the variable X_u, for a stochastic fact node u, is a Bernoulli random variable. For the node `deviceOnline(H,Platform)`, $P[X_u = 1]$ is the probability of the event that the device is online.

Using our success measurement model we accurately capture the effect of mobile devices as part of the network. Our evaluations in Section 7, quantifies the vulnerabilities introduced by mobile devices in an organizational network. We believe this analysis is valuable in order to make better decisions on the policies that determine mobile interactions with the network.

6 Determination of Initial Belief

In this section we describe the techniques and concrete examples for choosing initial belief values for fact nodes and improvement nodes.

Initial belief for fact nodes. An *initial belief value* is a given probability of success, $P(X_{u_i} = 1)$, at a fact node $u_i \in N_f$. Our success measurement model relies on a relatively small set of initial beliefs that provide an *estimation* of expected chance of success for specific attacks on network services. In an attack graph, these network service vulnerabilities are formalized as fact nodes. The methods for obtaining initial belief values may vary. We illustrate some specific approaches next.

For documented software vulnerabilities, the value of standard vulnerability scores (such as CVSS) is be used as an estimation of the expected chance of success of exploiting the vulnerability. The steps for assigning the initial belief values follows.

Analyzing the network configuration. A server A runs MySQL listening on port 3306, allowing remote connections. To protect A, iptables rules are set to allow tcp/udp connections either locally or to specific IP addresses inside a NAT subnet. These IP addresses belong to workstations from which the database administrators and developers connect to the server A, and a web server that runs the web applications.

Analyzing attacks and vulnerabilities. An attacker can exploit a remote privilege escalation vulnerability from a workstation W_1 to a developer workstation W_2. Since A accepts MySQL connections from W_2, the attacker uses one of multiple remote denial of service vulnerabilities (such as CVE-2012-3147, with a CVSS base score of $6.4/10$) to launch a denial of service on the MySQL server in A.

Assigning initial belief values. With multiple documented vulnerabilities with similar effects on u_2, we compute the value $P(X_{u_2} = 1) = max(s_1, s_2, \cdots, s_K)$, where s_j is a value in $[0, 1]$ based on the CVSS base score for a vulnerability j (for example, the

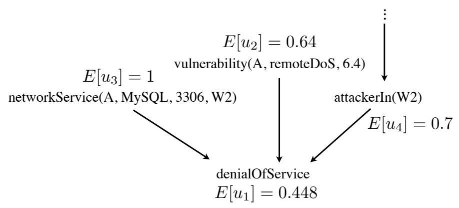

Fig. 5. u_1 is a denial of service on A, u_2 is a vulnerability, u_3 is a network service info, and u_4 indicates attacker reached W_2 that can access A.

score divided by 10), with K number of documented vulnerabilities. The computation of $P(X_{u_2} = 1)$ can be done in alternative ways, such as $P(X_{u_2} = 1) = \mu(s_1, s_2, \cdots, s_K)$, where μ is the mean of the score values.

We create another fact node as a dependency of the rule node u_1, denoted u_3, to indicate that incoming traffic on port 3306 is allowed from host W_2. We choose the probability value $P(X_{u_3} = 1) = 1$, indicating that the connection to the port 3306 is reliable and the attacker is knowledgeable about the port 3306 when attacking a MySQL database server. Otherwise, depending on the network configurations, we can set $P(X_{u_3} = 1) < 1$, with a reasonable value. The example rule node u_1 is shown in Figure 5.

Initial belief for improvement nodes. Initial belief values for improvement nodes correspond to the reliability of the security solution represented by the nodes. There are several assessment factors for computing the initial belief values. We categorize these factors into two main groups: (i) effectiveness and (ii) deployment. Effectiveness is measured by detection accuracy and the rate of false positive/negative decisions. The deployment factor includes measurements for memory consumption, CPU utilization, library dependencies, maintenance, and financial cost.

To compute an estimated initial belief value for a security product, we use a weighted function of effectiveness and deployment parameters. Let $Z_k^{(u_i)}$ a Bernoulli variable for an assessment factor u_i, and let L be the total number of assessment factors. We define the expected value for X_{u_i} as

$$E[X_{u_i}] = \frac{\sum_k E[Z_k^{(u_i)}]}{L}. \tag{9}$$

For an effectiveness factor k, the value of $E[Z_k^{(u_i)}]$ indicates the accuracy of improvement option u_i. For a deployment factor l, a higher value of $E[Z_l^{(u_i)}]$ indicates lower deployment overhead.

In the example scenario of Section 6, we create an improvement node for additional iptables rules to improve security. For instance, we modify the firewall rules on server A to allow connection to the database server on an unusual port p other than the default 3306, and also change MySQL socket configuration to listen on port p. Then we create an improvement node u_5 for an iptables rule dropping ICMP requests and limiting TCP ACK packets to already established connections to prevent the attacker from easily finding the port number p through a port scanner such as nmap. We expect that the firewall rule of the node u_5 has an average effectiveness (some attacks may bypass this rule) with virtually no deployment overheads. Thus, we compute the initial belief value for u_5 as $P(X_{u_5} = 1) = 0.5 * E[Z_1^{(u_5)}] + 0.5 * E[Z_2^{(u_5)}]$ with a value of $E[Z_1^{(u_5)}] \geq 0.5$ for the effectiveness factor and $E[Z_2^{(u_5)}] = 1$ for the deployment factor.

7 Experiments

In this section we present several experiments to verify the mathematical models introduced in Sections 4.2 and 5.1. We compute ECSA values for a network configuration obtained from a functioning real world network. The goal of our experiments is to demonstrate that the computation of our mathematical programming model is feasible. The results of our experiments are complementary to the established theoretical foundations of the sequential linear programming (discussed in Section 4.2.1).

We implemented the two computational methods and the security improvement model in Java (approximately 3500 lines of code). We use GLPK [20] for solving linear programs. Our implementation parses an attack graph input file (obtained from MulVAL [19]) and an initial belief file, computes the ECSA values according to various parameters, and performs security improvement analysis based on a set of improvement options and constraints.

In the following we describe our results for computing ECSA values, assess improvement options on our example network, and discuss the effect of mobile devices in the network. Our computation is efficient and scalable and runs in polynomial time.

The network data used for our experiments is collected from a functioning real world dynamic network of an organization (depicted in Figure 6) that serves a large number of users. The network has low usage restrictions and allows untrusted mobile devices to enter the network without mandatory security scanning[1]. In this network, a connected user can easily obtain information about the network topology, perform port scanning and operating system finger printing.

In the remainder of this section, we describe our experimental goals, analyze the results of the computation of ECSA for the network of Figure 6, and subsequently evaluate the security improvement options available for the example network.

1. Some of the data is sanitized yet preserves the general structure and vulnerability information.

7.1 Experimental setup

Our experiments are based on network configuration and vulnerability data from a real dynamic organizational network that is open to a large number of users and contains several servers and workstations. To simplify the discussion, our experiments include a subset of machines in the network.

We used network security scanning tools (such as nmap), online vulnerability repositories, and information provided by system administrators to create a network topology (depicted in Figure 6) and the attack graphs that represent the real network. We performed wireless network scanning to confirm the connectivity of wireless devices in the network. To assign initial belief values for the fact nodes, we followed the steps discussed in Section 6. Our program scans vulnerability score values and configuration settings to compute the initial belief values for computing the ECSA values.

In the following sections we describe four sets of experiments for which we generate two attack graphs (Table 1) with slight variations. We generate attack graph A (483 nodes) with no mobile devices in the network (i.e., availability of mobile devices is 0%) and attack graph B (549 nodes) that includes attack scenarios from untrusted mobile devices. Our experiments are computing ECSA for the attack graphs, computing an optimal placement for an intrusion prevention system, assessing the effect of mobile devices, and computing a network reconfiguration to minimize ECSA.

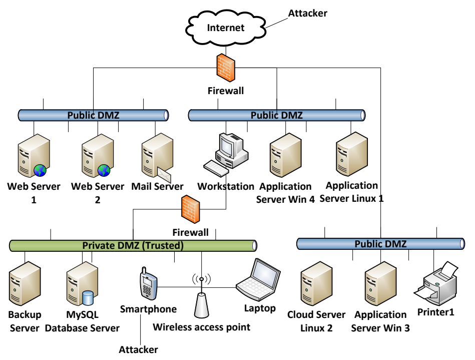

Fig. 6. Each machine on the three public DMZ subnetworks runs at least a network service with an open port. Data servers are on a NAT subnetwork and can only be accessed through the workstation. The attacker either attacks remotely or uses a phone to crack the wireless password and attack the servers.

For our experiments, we did not have access to data from attack incidents such as system logs to perform a validation of our results.

For the network architecture of Figure 6, we generate three attack graphs (Table 1) with slight variations. We generate attack graph A (483 nodes) with no mobile devices in the network (i.e., availability of mobile devices is 0%) and attack graph B (549 nodes) with possible attack scenarios from untrusted mobile devices.

Recall that a mobile device is defined to be a connected machine to the network that has a high degree of mobility and often joins other untrusted networks, thus making it exposed to a higher magnitude of infections that may in turn affect the security of the corporate network.

Attack Graph	Hosts	Nodes	Edges	Placement Options
A: No mobile	13	483	663	206
B: With mobile	13	549	757	235

TABLE 1
Attack graph A is generated with no mobile devices in the network and attack graph B is generated with two mobile devices. Placement options refers to the nodes that can be considered for the addition of an improvement node.

As described in Section 4.2.1, manual analysis of attack graphs A, B, and C requires several iterations of computation according to the sequential linear programming. Moreover, by manually inspecting the network, through configurations, network topology, and software vulnerabilities, it is difficult to find reasonable conclusions about chances of successful attacks or the effect of particular devices on the network.

The goal of this experiment is to evaluate the results of computing the ECSA values for attack graph A. In this experiment, we compute the ECSA values based on the maximization approach to find the most vulnerable network components as described in Section 4.2.1. Such a computation can directly influence the decisions made by a system administrator. That is, knowing which network components may cause higher threats to the network enables system administrators to consider hardening the network security by addressing the issues presented in the most vulnerable components.

7.2 Chances of a Successful Attack

Using the maximization method (Section 4.2.1), we performed an experiment on attack graph A (with no mobile devices) for which the database server is the attack's target. A sample of our results is depicted in Figure 7. The x-axis shows the targets of exploitation goal nodes achievable by the attacker. Notice that the maximization method computes the maximum ECSA values for the nodes of the attack graph. The maximized ECSA values show which network components (e.g., servers) are the most vulnerable.

The results of this experiment suggest that both application servers 3 and 4 (denoted server3 and server4 in Figure 7) have high ECSA values for their goal node, indicating high chances of successful attacks. This is because application servers 3 and 4 have highly scored software vulnerabilities, a high number of open ports, and thus they are relatively more exposed to the outside world. However, the chances of successful attacks on the target database server is the lowest, which is due to a better network configuration to protect it. This result is expected as the database server is less exposed to the outside and runs fewer vulnerable network services.

Figure 7 also shows the results for the ECSA computed based on attack graph B, which are discussed in Section 7.4.

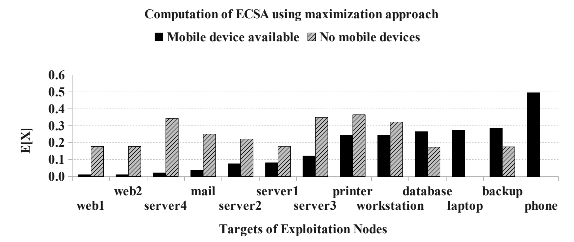

Fig. 7. ECSA values attack graph A (no mobile devices) and B (with mobile devices). In the experiment with mobile devices, the availability of a mobile device is captured with a random variable and is not assumed to be fixed.

7.3 Optimal Placement of Security Products

We tested our security improvement method introduced in Section 5.1 with a single improvement applied to a comprehensive set of rules in attack graph B. We used the results from the previous section to find the best placement of an improvement option for the network of Figure 6. Our improvement option is the installation of an intrusion prevention system (IPS) on a single device to minimize the risk on the target host (the database server). Our choice of IPS has some deployment overhead because of memory and CPU usage. After testing its effectiveness, we believe that this IPS has a low false negative rate. Using Equation 9, the initial belief for each improvement fact node for the IPS is $E[X_{\mathcal{T}}] = 0.3$.

This assumption can be modified with no effect on the original model. This experiment is performed on attack graphs A and B.

According to our method (described in Section 5.1), we add all the exploitation rules, to the set of applicable placement nodes N_{ra} (i.e., 206 nodes for attack graph A and 235 nodes for attack graph B[2]). Then we modify the original attack graphs to include improvement fact nodes as predecessors to each $u \in N_{ra}$.

2. Note that one can choose fewer rule nodes for solving the optimal placement problem, depending on possible placement constraints.

A quick reasoning may recommend that the target server (i.e., database server) itself must be where we install the IPS. However, this recommendation may not be optimal. We computed the improvement for the attack graph with no mobile devices and with the mobile devices present in the network. Table 2 shows the improvement results, for each attack graph configuration, ordered based on the percentage decrease in $E[X_{\mathcal{G}}]$. Third column shows the best placement of the IPS. $E'[X_{\mathcal{G}}]$ and $E[X_{\mathcal{G}}]$ denote the expected chances of a successful attack for $\mathcal{G}$ (i.e., the database server) in the improved attack graph and the original attack graph, respectively.

Rank	Attack Graph	Machine	$E'[X_{\mathcal{G}}]$	$E[X_{\mathcal{G}}]$	% $\downarrow$
	A: No mobile	App Server 3	0.0520	0.1739	70.09
1	B: With mobile	Database	0.0521	0.2651	70.04
	A: No mobile	Database	0.0552	0.1739	79.18
2	B: With mobile	Workstation	0.0791	0.2651	70.16

TABLE 2
Optimal selection for IPS installation for attack graphs A and B. The attack target is code execution on the database server. The results are compared against the original ECSA values without the improvement option. $E'[X_{\mathcal{G}}]$ is the ECSA value of the improved model.

The results of the Table 2 demonstrate significant decrease in $E[X_{\mathcal{G}}]$ when considering the improvement option for attack graphs A and B. Our results indicate that installing the IPS on application server 3 has the best effect in minimizing the ECSA of the attack's goal. The reason is that the target server can be attacked from a number of ports indicated by goal nodes. Based on the computed values of the random selectors Y_i, a particular port p_1 receives a high chance of attacking the database server.

In the results, attacking the database server from p_1 has a lower ECSA compared to attacking application server 3. In the attack graph, attacking application 3 is a predecessor of attacking the database server on port p_1. Thus, the improvement option multiplied with ECSA of attacking application server 3 reduces the value of $E[X_{\mathcal{G}}]$ more, and installing the IPS on application server 3 yields a slightly lower value of $E[X_{\mathcal{G}}]$.

Notice that the second ranked improvement recommendation suggest the workstation as the best place to install the IPS. This is consistent with the conclusions from the ECSA values since the workstation is one of the most vulnerable devices determined in the previous experiment.

7.4 Effect of Mobile Devices

The network architecture presented in Figure 6 is also vulnerable to threats from mobile devices. For example, in the network of Figure 6, the system administrators have allowed mobile devices to join the wireless access point that is set up for internal purposes in the private DMZ region. Also, the laptop (connected to the wireless access point) is directly accessible from the workstation and the printer. Such configurations increase the attack surface. We assessed the security of the network by computing the ECSA on attack graph B that includes the attack vectors from mobile devices.

The ECSA in our experiments is computed according to the method for computing the most vulnerable components (Section 4.2.1). Therefore, the results of the experiment on attack graph B (Figure 7) show lower values for exploiting the application servers, but higher values for exploiting the smartphone and the laptop (with high scored known software vulnerabilities), the workstation, and the printer. This is because the mobile devices in the network of Figure 6 have highly scored vulnerabilities that make them more attractive to attackers.

From the results of the experiment with mobile devices, we can conclude that the presence of highly vulnerable mobile devices in the network increases the chance of attack on the target machine. Using attack graph B, the most vulnerable components are the workstation, the printer (which has vulnerable server software), and the mobile devices (i.e., the laptop and the smartphone). In this experiment, the chance of success on exploiting the database server is increased by 52.44%.

7.5 Improving Network Configuration

Our optimal recommendation method is capable to compute an improved network configuration with no extra security products (such as an IPS) added to the network. In particular, we find a port p (amongst all open ports on all machines) such that if it is disabled, the value of $E[X_{\mathcal{G}}]$ is minimized. That is, for any other port p', if p' is disabled in the network (for which we obtain $E'[X_{\mathcal{G}}]$), then $E'[X_{\mathcal{G}}] \geq E[X_{\mathcal{G}}]$.

We used our method to examine the option on every possible open port that appears in the attack graph. The results of our experiments on attack graphs A (no mobile) and B (with mobile) are summarized in Table 3.

Rank	Attack Graph	Machine, Port	$E'[X_{\mathcal{G}}]$	$E[X_{\mathcal{G}}]$	$\% \downarrow$
1	A: No mobile	Database, 2200	0.0	0.1739	100
	B: With mobile	Database, 2200	0.0	0.2651	100
2	A: No mobile	App Server 3, 22	0.0	0.1739	100
	B: With mobile	Backup, 2200	0.12	0.2651	53.8

TABLE 3
Optimal selection for closing a single port with the best effect on the security of the network.

To verify the accuracy of our method, we considered open ports on the target database server that if disabled would eliminate the chance of attack. Although it is a common practice to eliminate straightforward attacks on well known ports, some of the servers in the target network did have open ports with minimum firewall rules.

The results in Table 3 show that the best recommendation is to disable the port 2200 and that would achieve a zero expected chance of successful attack. The second ranked recommendations are to close ports on the application server 3 and the backup server. Notice that both recommendations achieved a lower value of $E[X_{\mathcal{G}}]$, thus improving the security of the network.

8 Conclusions and Future Work

In this work we formalized, implemented, and evaluated a new probabilistic model for measuring the security threats in large enterprise networks. The novelty of our work is our ability to quantitatively analyze the chances of successful attacks in the presence of uncertainties about the configuration of a dynamic network and routes of potential attacks. To demonstrate the importance of our technique, we showed the use of our success measurement in solving two open problems in network security: *i)* How to optimally deploy security products and services across the network? *ii)* How to formally analyze the vulnerability of dynamic networks with mobile devices?

For the future work, we plan to utilize and extend our success measurement model and optimal security placement algorithm to solve more complex network security optimization problems.

References

[1] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Computer Security Applications Conference, 2006.*, 12 2006, pp. 121 –130.

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

[3] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*, 2002, pp. 49–63.

[4] S. Fenz, "An ontology- and Bayesian-based approach for determining threat probabilities," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 344–354.

[5] 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*, ser. QoP '08. New York, NY, USA: ACM, 2008, pp. 23–30. [Online]. Available: http://doi.acm.org/10.1145/1456362.1456368

[6] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, Jan 2012.

[7] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using Bayesian networks for cyber security analysis," in *The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2010.

[8] S. Noel, S. Jajodia, L. Wang, and A. Singhal, "Measuring security risk of networks using attack graphs," *International Journal of Next-Generation Computing*, vol. 1, no. 1, July 2010.

[9] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Proceedings of the 22^{nd} annual IFIP WG 11.3 Working Conference on Data and Applications Security*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 283–296.

[10] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing, "Ranking Attack Graphs," in *Recent Advances in Intrusion Detection*. Springer Berlin, 2006, vol. 4219, pp. 127–144.

[11] R. E. Sawilla and X. Ou, "Identifying Critical Attack Assets in Dependency Attack Graphs," in *Proceedings of the 13^{th} European Symposium on Research in Computer Security: Computer Security*, ser. ESORICS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 18–34.

[12] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*, june 2012, pp. 1–12.

[13] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *Proceedings of the 14^{th} ACM conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 204–213.

[14] S. Noel and S. Jajodia, "Optimal IDS sensor placement and alert prioritization using attack graphs," *Journal of Network and Systems Management*, vol. 16, no. 3, pp. 259–275, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10922-008-9109-x

[15] P. Mell, K. Scarfone, and S. Romanosky, "Common Vulnerability Scoring System," *IEEE Security & Privacy*, vol. 4, no. 6, pp. 85–89, 11 2006.

[16] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated Generation and Analysis of Attack Graphs," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=829514.830526

[17] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming*. John Wiley and Sons, Inc., 2005.

[18] F. Palacios-Gomez, L. Lasdon, and M. Engquist, "Nonlinear optimization by successive linear programming," *Management Science*, vol. 28, no. 10, pp. 1106–1120, 1982. [Online]. Available: http://www.jstor.org/stable/2630940

[19] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13^{th} ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 336–345. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180446

[20] R. Ceron, "The GNU linear programming kit, part 1: Introduction to linear optimization," 2006, http://www.ibm.com/developerworks/linux/library/l-glpk1/.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., September 1999.

[22] Z. Zhang, F. Naït-Abdesselam, X. Lin, and P.-H. Ho, "A Model-based Semi-quantitative Approach for Evaluating Security of Enterprise Networks," in *Proceedings of the 2008 ACM Symposium on Applied Computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 1069–1074.

[23] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A weakest-adversary security metric for network configuration security analysis," in *Proceedings of the 2^{nd} ACM Workshop on Quality of Protection*, ser. QoP '06. New York, NY, USA: ACM, 2006, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/1179494.1179502

[24] L. Wang, A. Singhal, and S. Jajodia, "Measuring the overall security of network configurations using attack graphs," in *Proceedings of the 21^{st} annual IFIP WG 11.3 working conference on Data and applications security*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 98–112. [Online]. Available: http://dl.acm.org/citation.cfm?id=1770560.1770573

[25] S. Bhatt, W. Horne, and P. Rao, "On computing enterprise IT risk metrics," in *Future Challenges in Security and Privacy for Academia and Industry*, ser. IFIP Advances in Information and Communication Technology. Springer Boston, 2011, vol. 354, pp. 271–280.

[26] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Computer Communications*, vol. 29, no. 18, pp. 3812–3824, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2006.06.018

[27] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz, "On the convergence of successive linear-quadratic programming algorithms," *SIAM J. on Optimization*, vol. 16, no. 2, pp. 471–489, Jun. 2005.

[28] R. Fletcher, N. I. M. Gould, S. Leyffer, P. L. Toint, and A. Wächter, "Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming," *SIAM J. on Optimization*, vol. 13, no. 3, pp. 635–659, Aug. 2002.

[29] 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 27^{th} Annual Computer Security Applications Conference*. New York, NY, USA: ACM, 2011, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076738

[30] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. Spafford, "ADEPTS: adaptive intrusion response using attack graphs in an e-commerce environment," in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, 2005, pp. 508–517.