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

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

Abstract—Securing the networks of large organizations is technically challenging due to the complex configurations and constraints. Managing these networks requires 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. In this paper, we address the problem of statically performing a rigorous assessment of a set of network security defense strategies with the goal of reducing the probability of a successful large-scale attack in a dynamically changing and complex network architecture. We describe a probabilistic graph model and algorithms for analyzing the security of complex networks with the ultimate goal of reducing the probability of successful attacks. Our model naturally utilizes a scalable state-of-the-art optimization technique called sequential linear programming that is extensively applied and studied in various engineering problems. In comparison to related solutions on attack graphs, our probabilistic model provides mechanisms for expressing uncertainties in network configurations, which is not reported elsewhere. We have performed comprehensive experimental validation with real-world network configuration data of a sizable organization.

Index Terms—Network security, attack graph, probabilistic model, vulnerability analysis, optimization

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 systems (IPSs).

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 for a few attempts [6], [12], [13], [14], rigorous techniques for risk reduction have not been reported.

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 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 standard technique called sequential linear programming (SLP). Our

• *H.M.J. Almohri is with the Department of Computer Science, Kuwait University, Kuwait. E-mail: almohri@cs.ku.edu.kw.*
• *L.T. Watson is with the Departments of Computer Science and Mathematics, Virginia Tech, Blacksburg, VA 24060. E-mail: ltw@cs.vt.edu.*
• *D. Yao is with the Department of Computer Science, Virginia Tech, Blacksburg, VA, 24060. E-mail: danfeng@cs.vt.edu.*
• *X. Ou is with the Department of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506. E-mail: xou@ksu.edu.*

Manuscript received 16 Dec. 2013; revised 31 Jan. 2015; accepted 3 Mar. 2015. Date of publication 9 Mar. 2015; date of current version 13 July 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2015.2411264

algorithm is proved to converge and it is scalable to large networks with thousands of components and 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.
- *An efficient security optimization model*, generated based on a quantified attack graph, to compute 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 probability 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. Our implementation efficiently computes the probabilities throughout large attack graphs with a quadratic execution performance.

2 Related Work

The literature has a significant number of attempts to provide methods, algorithms, and tools for the various problems concerning graph-based analysis of security in large networks. Graph-based analysis of networks was proposed in [15] where a graph of attack stages in a network topology was introduced to analyze specific attacks in a network. The work in [15] was followed by the method proposed in [16] that in addition to producing attack graphs using model checking, introduced an analysis of guarding options against the attacks.

The effort to enhance graph-based analysis and security hardening has continued since [15] and [16]. Unfortunately, some of the ongoing challenges facing automated network security analysis remain unresolved. Per our survey, the literature lacks a comprehensive and rigorous methodology for the assessment of a set of network security defense strategies with the goal of reducing the success of an attack. In the following, we present a thorough comparison of our work with the related research followed by a summary of the novelty of our work.

2.1 Probabilistic Analysis

Using the probability theory to compute a quantitative security has been reported in [4], [5], [9], [17]. For example, Wang et al. [9] designed a probabilistic model for computing a security risk metric using attack graphs. A recent work models the behaviors of complex cognitive radio software with hidden Markov models [17].

Bayesian analysis of networks using attack graphs [5], [6], [7] differs from 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. Bayesian threat probability based on security and organization-specific knowledge as well as attacker profile is discussed in [4]. 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, which is addressed in our work.

The work in [18] attempts to broaden the definition of attack graphs as well as providing algorithms for predicting vulnerabilities in the network. This work introduces temporal probabilistic attack graphs that are used to update the vulnerability information in time. Our work differs with [18] in the modeling assumption. The probabilistic attack graph presented in [18] models time intervals for each attack step that may or may not occur with specific probabilities. While this is a very useful approach, our work models a direct attack step probability based on the Bernoulli model of successes and failures considering uncertainties in an attacker's action and a notion of device availability (i.e., assuming a device may not always be connected to the network). Despite that we share the same goal of *mitigating security risks*, our work takes a different approach through the application of efficient mathematical programming methods to security optimization problems that we point out in Section 3.

The work in [9] discusses an interpretation of the metric and a heuristic to compute the metric. In our work, we provide a success measurement model that generalizes the method in [9] by capturing the uncertainty in attacker's choices (discussed as a random selector in Section 4).

2.2 Ranking

In attack graph ranking, an initial input score is used to bootstrap a ranking algorithm that produces a quantified attack graph. A number of attack graph ranking algorithms are inspired by and extensions of PageRank [19]. PageRank is an algorithm proposed by Page et al. [19], which is used to rank important webpages.

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 [20], a weakest-adversary approach to ranking attack graphs [21], a generic framework for an attack resistance metric [22], and an enterprise IT risk metric using CVSS scores [23].

2.3 Security Improvement

Quantified attack graphs or similar formalism are particularly useful when utilized as a basis for improving the security of a network. The authors in [6], [13] proposed solutions for the security hardening problem as a multiobjective optimization problem. The main advantage of our work compared to the use of genetic algorithms in [13] is that we formulate the security hardening problem as a general mathematical programming problem that is directly developed according to an attack graph. The mathematical programming problem presented in this paper can be extended to consider a variety of constraints that we discuss as a future direction. Moreover, our work differs in the research goal as we focus on reducing the success rates of attackers, whereas the work in [13] is on optimizing costs (similar to [12]) and reducing damages. Noel and Jajodia presented a greedy solution for the problem of the best placement of IDS sensors in a network using attack graphs [14]. The solution finds a minimal number of sensors that can cover all critical attack paths. Wang et al. proposed a method for finding the (initial) conditions that need to be removed to improve network security [24]. Both these solutions aim at reorganizing the networks to improve security. In comparison, our work provides network hardening solutions beyond network reorganization. Our probabilistic model supports the computation of optimal network security defense strategies.

Huang et al. proposed a method for distilling the critical attack graph surface iteratively through minimum-cost SAT solving [25]. 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], a probabilistic metric was introduced. The core component of the proposed work is to simulate the attack scenario and provide recommendation options to find a better configuration of the network. Comparably, our improvement model is not limited to making an optimal choice between available configuration options. Our work goes further by considering additional security hardening options (such as installing an IPS) and finding an optimal recommendation accordingly. Our proposed model finds an optimal recommendation based on a nonlinear program and is not limited to simulation results. 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 availability of machines and uncertainty in attackers' decisions.

2.4 Summary of Comparisons

In summary, there are several differences that distinguish our work from the existing research.

1. 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 availability, and the effect of mobile devices.
2. 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.
3. 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

Motivated by the general research goal of developing optimized network security settings, our work focuses on the problem of *statically performing a rigorous assessment of a set of network security defense strategies with the goal of reducing the probability of a successful large-scale attack in a dynamically changing and complex network architecture*. This problem represents a practical concern in modern organizational networks, where there is a need for a highly reliable and mathematically sound platform to conduct effective security hardening analyses.

3.1 Challenges

In addressing the aforementioned problem, our work faces and attempts to solve several technical challenges. First, to provide a reliable analysis for improving network security, we face the challenge of developing a rigorous model that accurately captures the reality. Though this is not an entirely new challenge, the current state-of-the-art does not focus on the rigorousness of the proposed models. As explained in Section 4, we approach this challenge by developing a model with a clear theoretical foundation that accurately captures a complete view of a multistage attack on a network. The main advantage of our model is in the use of established mathematical concepts that best fit the problem, which enables us to exploit efficient methods to develop reliable algorithms.

Second, most of the literature in attack graph analysis focuses on various techniques to transform what we call plain attack graphs (that is, attack graphs with no quantification metrics) to quantified attack graphs that provide clearer insights into the seriousness of the attacks. Effective assessment of network security defense strategies remains a challenge that requires significant effort in terms of further enriching the attack graph model and transforming it into an *identical* optimization problem. We address this challenge by developing theoretically sound mathematical models that represent a complete view of attack graphs, and are capable of including candidate network security defense options. The result of our optimization is to find which network security defense strategy will yield enhanced security according to the provided input.

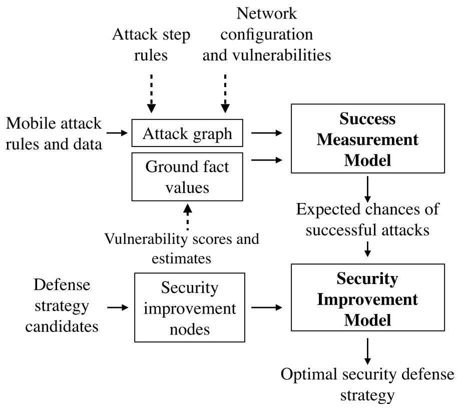

Attack step rules
Network configuration and vulnerabilities
Mobile attack rules and data
Attack graph
Ground fact values
Success Measurement Model
Expected chances of successful attacks
Vulnerability scores and estimates
Defense strategy candidates
Security improvement nodes
Security Improvement Model
Optimal security defense strategy

Fig. 1. Our models work based on three input sets from attack graph generators as well as initial belief values associated with potential vulnerabilities and network configuration data.

For example, a system administrator managing a complex network architecture can use our method to compare and contrast the effectiveness of two security defense packages, one to install a software firewall on a number of hosts, thus downgrading computational performance and potentially increasing false positives, and the other to move a set of services behind a hardware-based load balancer and thus increasing cost as well as network latency.

Third, we discover and address a novel challenge in systematically modeling the *uncertainties in an adversary's attack steps towards a major attack goal*. This is a problem when dealing with attacks of multiple steps. For instance, an adversary may be faced with a range of vulnerabilities to try to exploit when executing an attack step. When analyzing the level of security in a network, in the lack of historical attack data, it is particularly challenging to deal with such uncertainties at the modeling level. We use a statistical approach where we define a random behavior to model the various possibilities of attacks. Specifically, we define a special random variable Y_{u_i} for each possible attack step within an attack path. The value of Y_{u_i} corresponds to the probability that the adversary chooses an attack step u_i. We further explain the details of defining and using this method in Section 4.1.

3.2 Approach

We approach the challenges mentioned in Section 3.1 by defining, implementing, and experimenting with a new probabilistic quantification model that we combine with our novel optimization problem as described in Sections 4 and 5. Our probabilistic quantification model, referred to as *success measurement model*, quantifies the vulnerabilities of networked components and resources, by computing the expected chance of successful attack (ECSA) at *every attack step*, which is represented by an attack graph node. Our *security improvement model* uses the computed probabilities from the success measurement model to find optimal security defense strategies given a set of available options.

As depicted in Fig. 1, the computation in the success measurement model requires three sets of inputs, which are a set of attack steps, a set of network configuration and potential vulnerabilities, and a set of ground facts. The first set includes the steps necessary to execute a targeted attack in a network. These steps represent intermediate attack goals such as compromising a machine that has an internal connectivity with a targeted server. In addition, the attack steps also describe the various parallel choices available to an attack when achieving a specific target. The second set includes the network configurations and vulnerability data that collectively provide host software installations, inter host connectivity, running services and connections, and known or potential software vulnerabilities. The third set contains the ground fact values that describe the vulnerability, availability, and connectivity of various network configuration.

In our implementation, the first two sets of inputs (i.e., the attack steps and the network configuration data) are taken from dependency attack graphs. The system administrators use vulnerability assessment tools (such as OVAL [26]) to explore the configurations and vulnerability data in their networks. The output of such assessment is provided as an input to attack graph generation tools. Attack graph generation tools (such as MulVAL [27]) often include customized predefined attack step rules that are applied to the configurations and vulnerability data of a network and produce a plain (that is, not quantified) attack graph. The additional step required by our model is to develop a set of ground fact values (described in detail in Section 6). The values bootstrap the computation of success probabilities throughout an attack graph.

The output of the computation based on our success measurement model is the input to the security optimization model (Fig. 1). Using the security improvement model, we transform the quantified attack graph from the success measurement model into a mathematical program. The resulting mathematical program includes an additional set of data that represent various network security defense strategies. In the tool that we developed, the security administrators simply feed this information as logical predicates such as `ips_installed(T, E)`, which describes a potential installation of an intrusion prevention system of type `T` and security effectiveness `E`. The effectiveness value `E` is a score estimated by the system administrator based on prior experiences and available effectiveness data.

3.3 Results

Validating the results of theoretical modeling of network security under the assumption of *lack of data* is challenging. In this work, we only use the data from attack graphs to perform a manual analysis of the results produced from the application of our two models. We set up our experiments based on the network configuration data, existing potential vulnerabilities, and attack graphs produced for a functioning real world corporate network. We summarize our experience with implementing the models as follows:

1. All the algorithms were programmed from scratch in Java, automating the entire process for receiving input from attack graph generators until recommending the best security defense strategies.
2. The implementation performance only relies on the performance of the simplex method used for solving the optimization problem. Since the simplex method is heavily and successfully used in practice [28], our

model features a high level of computational scalability and efficiency.

In addition, we give a summary of our experiments (Section 7) next.

1. The focus of our experiments is to practically demonstrate the practicality, feasibility, and accuracy of the model.
2. Our experiments include novel features such as analyzing networks with less studied but potentially vulnerable devices such as mobile devices and networked printers. To the best of our knowledge, the experiments in the network analysis literature lack this level of detail.
3. Our model will give system administrators a solid analysis of the security in their networks that will assist in actual implementation of security features to downgrade the possibility of successful attack.

4 Success Measurement Model

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

In a logical attack graph, nodes are of three types and are defined as tuples.

Definition 2. *Each attack graph node u is a tuple $(d_u, E[X_u])$ where d_u is the description of a network configuration item (when $u \in N_f$), an attack rule (when $u \in N_r$), or an attack goal (when $u \in N_g$), and $E[X_u] \in [0, 1]$ is the corresponding ECSA (see (1) and (3)) of the node u.*

A rule node in an attack graph represents a logical conjunction of its predecessors, a goal node in an attack graph represents a logical disjunction of its predecessors, and a fact node is a node with no predecessor.

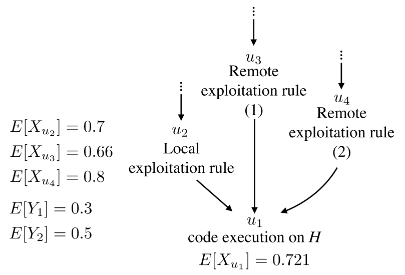

$E[X_{u_2}] = 0.7$
$E[X_{u_3}] = 0.66$
$E[X_{u_4}] = 0.8$
$E[Y_1] = 0.3$
$E[Y_2] = 0.5$
u_3 Remote exploitation rule (1)
u_4 Remote exploitation rule (2)
u_2 Local exploitation rule
u_1 code execution on H
$E[X_{u_1}] = 0.721$

Fig. 2. 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).

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 3. *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 $\phi(u) = \{v_1, v_2, \ldots, v_t\}$. 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 4.4).

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 Fig. 2.

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.

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 these choices.

In the the attack graph of Fig. 2, 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.

Definition 4. *A random selector Y_i is a Bernoulli variable that is associated with a rule node u_i. Y_i acts as a weighting variable for the corresponding rule node variable X_{u_i}. For any goal node v, with a set of predecessor rule nodes $\phi(v)$, we have $\sum_{u_i \in \phi(v)} E[Y_i] = 1$.*

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 $\phi(u) = \{v_1, v_2, \ldots, v_t\}$ 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

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

Observe that the definition above selects $X_u = X_{v_i}$ by the event $Y_i = 1$, $Y_j = 0$ for $j < i < t$ (for example, Fig. 2). 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.

4.2 Computing ECSA Values

From a defender's point of view, attack choices are uncertain with various attack scenarios. Existing work such as [9], [11], [16], has 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 chance of success for an attack.

Finding 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 represent the nodes of an attack graph.

Let $x_i = E[X_i]$ be a decision variable for a node $i \in N_r \cup N_g$, and $x = (x_1, x_2, \ldots, x_M)^T$ be the vector of unknown ECSA values for all nodes. Let $y_i = E[Y_i]$ be a decision variable for a random selector Y_i, and $y = (y_1, y_2, \ldots, y_P)^T$ 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

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

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 $\phi(u) = \{v_1, v_2, \ldots, v_t\}$, 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 $y_{m+1}, y_{m+2}, \ldots, y_{m+t-1}$ for some $m = m_u$ depending on u. Note that there is no variable y_{m+t} since y_{m+t} is dependent on $y_{m+1}, y_{m+2}, \ldots, y_{m+t-1}$; $y_{m+t} = 1$ only when all other selectors for u are zero.

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

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

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) set to zero.

4.3 Computational Procedure

For a network configuration w, let G_w be the corresponding attack graph. The complete procedure to compute the ECSA values of nodes (Definition 2) for an attack graph (Definition 1) is given next.

To prepare the attack graph for computation, we execute the following procedure.

Procedure 1:

1. Determine the set of initial belief values $B^0 = \{E[X_{u_1^f}], E[X_{u_2^f}], \ldots, E[X_{u_{|N_f|}^f}]\}$ for each fact node $u^f \in N_f$. Let B_i^0 denote $E[X_{u_i^b}]$.
2. Create a set of fact nodes N_f' such that $|N_f'| = |N_f|$, where for each node $u_i \in N_f$ there is a node $v_i \in N_f'$ corresponding to the same network item description (i.e., d_{u_i} is identical to d_{v_i}) and with $E[X_{v_i}] = B_i^0$.
3. Update the attack graph G_w such that the original set of fact nodes N_f is replaced with the new quantified set of fact nodes N_f', producing attack graph G_w'.

We input the resulting attack graph G'_w to the procedure below for computing the maximum possible ECSA values.

Procedure 2:

1. Transform G'_w into the corresponding mathematical program P_w as explained in Section 4.2.
2. With $Z = (x, y)$, choose a starting point Z^0 with each variable being a random value in the range $[0, 1]$.
3. Replace all the nonlinear functions $f_i(Z)$ with a linear approximation $f_i(Z) \approx f_i(Z^0) + \nabla f_i(Z^0)(Z - Z^0)$.
4. To prevent large changes in Z, add the constraint $|Z_i - Z_i^0| \leq \epsilon$, that is, each variable can change by no more than ϵ.
5. Solve the resulting LP problem using an efficient LP method, such as the simplex method, producing the candidate optimal point Z^*, which replaces Z^0.
6. Repeat step Steps 3-5 until the solution converges to a stationary point.

Procedure 2 computes the maximum possible ECSA value for node u in the attack graph. Our procedure is a technique called sequential linear programming [29]. SLP is a standard technique for solving nonlinear optimization problems, which is found to be computationally efficient and converges to an optimal solution [30].

Complexity. In terms of computational efficiency, all of the steps in Procedures 1 and 2 require polynomial time in the number of nodes. The most complex step is the fifth step in Procedure 2. The complexity of the fifth step depends on the complexity of the LP algorithm, and the simplex method is polynomial in practice [28]. Since SLP has linear convergence, the number of iterations is also polynomial.

Optimizing the initial attack graph. Our attack graphs have goal nodes with no outgoing arcs (the ultimate goal) and may include several paths towards satisfying a goal. It is possible to remove unnecessary paths that may or may not be taken by an attacker towards the ultimate goal. An experienced attacker may take a near minimum path towards the goal, thus, saving time and perhaps bypassing difficult paths along the way. Finding a minimal attack graph will precede any computation with regard to the ECSA value.

4.4 Attack Dependencies

A major problem in probabilistic risk assessment is to accurately capture attack step 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 computed by considering the dependency relations determined in an attack graph.

Definition 5. *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 5, 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 $X_v = 1$ indicates 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 attack. The attack step dependencies could occur when multiple consequent attacks are analyzed. To compute these dependencies, consider the following formulation.

Let A, B, and $C = AB$ be the random variables associated with nodes in an attack graph. If we assume A and B are dependent, then

$$\begin{aligned} E[C] &= P[C = 1] = P[A = 1 \wedge B = 1] \\ &= P[A = 1 | B = 1]P[B = 1] = E[A | B = 1]E[B]. \end{aligned}$$

To compute $E[A|B = 1]$, set $B = 1$, which forces all predecessors of B to also be 1, and recompute all expected values at nodes affected by assuming $B = 1$. After

$$E[C] = E[A|B = 1]E[B]$$

is computed, the rest of the computation proceeds without modification.

Given the above formulation, we conclude that considering dependence in attack information in our success measurement model (despite existing probabilistic work) is straightforward and does not require significant additional computation.

5 Security Optimization

To achieve our main research goal (described in Section 3) of reducing the probability of success in an attack, and thus optimizing the overall security of the network, we point out the necessity to model this problem as an optimization problem. Further, we attempt to model an important feature that is to consider the availability of machines in the network. In this section we describe these two contributions of our work as summarized below.

- *Optimizing the security of the network.* Given a set of security hardening products (e.g., a host based firewall), we compute an optimal distribution of these resources subject to given placement constraints. Using the rigorous probabilistic model introduced in Section 4.1, this is the first work in which a logical attack graph (Definition 1) is transformed into a system of linear and nonlinear equations with the global objective of reducing the probability of success on the graph's ultimate attack goal. This transformation is performed efficiently and naturally and directly captures our research goal.
- *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.

5.1 Optimizing the Security of the Network

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 the effects of various placements of one or more improvement nodes 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 $j_1 < j_2 < \cdots < j_{\mathcal{A}}$ 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

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

f_u is unmodified for rule nodes $u \in N_r$. 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

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

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

Fig. 3. The two predicates describe attack stages (i.e., remote and local exploits). The predicate deviceOnline(H,Platform) captures the availability of the device H.

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

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

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 Machine Availability and Threats from Mobile Devices

To capture the increase in 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 [31] 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 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. Accordingly, we define basic exploitation rules for mobile devices in Fig. 3.

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.

Note that our intuition of availability is in its general sense and is not necessarily bound to the availability as, for example, connectivity of a machine. An availability element in our method may capture a machine's connectivity, responsiveness of a particular vulnerable service, or the role of a firewall rule that might limit the availability of a service. For instance, our model captures the scenario in which a software firewall, based on a specific policy, limits the number of TCP connections opened by the Apache web server, and, for a specific period of time. When analyzing a vulnerable network according to our model, a fractional probability value on a fact node representing a service such as Apache will accurately imply limited attack chance as a result of the limited availability.

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. 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. Each stochastic fact node u is represented using a Bernoulli random variable X_u. For instance, for the node `deviceOnline(H,Platform)`, $E[X_u] = P[X_u = 1]$ is the probability of the event that the device is online.

6 Determination of Initial Belief

In this section we discuss and provide a concrete example for choosing initial belief values for fact nodes and improvement nodes.

6.1 Discussion

Our model relies on the availability of the initial belief values that are *initial estimates of vulnerabilities* at a subset of nodes in an attack graph. Since this raises a concern about the practicality of determining these values, there have been a number of recent attempts to automate this process. One notable attempt is presented by Wang et al. [32] where a vulnerability assessment metric is developed that can be computed regardless of the type of the vulnerability itself. That is, the metric relies on the number of unknown vulnerabilities required to compromise a network.

References [33] and [34] provide various ways to calculate metrics for zero-day or known vulnerabilities. While a comprehensive measurement model for computing initial estimates of vulnerability impacts may still be needed, existing methods as well as expert knowledge suffice for our computations.

6.2 Example

Initial belief for fact nodes. An *initial belief value* is a given probability of success $P(X_{u_i} = 1)$ at a fact node $u_i \in N_f$.

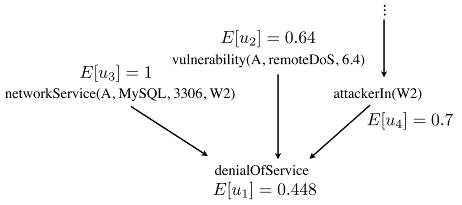


Fig. 4. u_1 is a denial of service on A, u_2 is a vulnerability, u_3 is a network service info, and u_4 indicates attacker reached W_2 that can access A.

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 used as an estimation of the expected chance of success in exploiting the vulnerability. The steps for assigning the initial belief values follow.

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 attack on the MySQL server in A.

Assigning initial belief values. With multiple documented vulnerabilities with similar effects on u_2, we compute the value $P(X_{u_2} = 1) = \max(s_1, s_2, \ldots, s_K)$, where s_j is a value in $[0, 1]$ based on the CVSS base score for a vulnerability j (for example, the score divided by 10), with K documented vulnerabilities. Alternatively take $P(X_{u_2} = 1) = \mu(s_1, s_2, \ldots, s_K)$, where μ is the mean of the score values.

We create another fact node as a dependency of the rule node u_1 (see Fig. 4), 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.

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

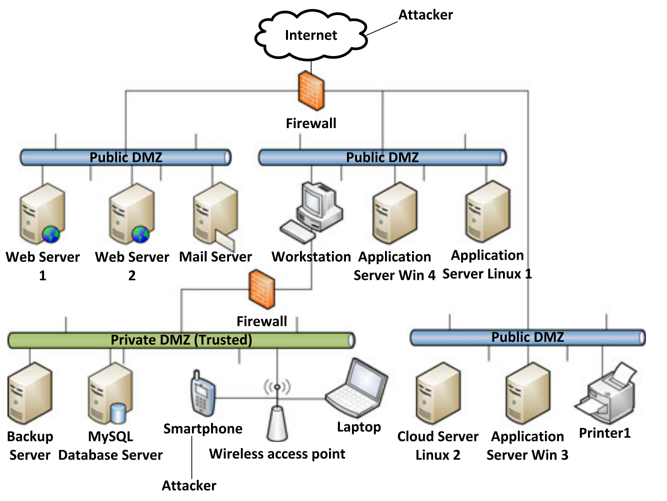


Fig. 5. Each machine on the three public DMZ subnetworks runs at least a network service with an open port. Data servers are on a NAT subnetwork and can only be accessed through the workstation. The attacker either attacks remotely or uses a phone to crack the wireless password and attack the servers.

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 the mean of all the effectiveness and deployment parameters. Let $Z_k^{(u_i)}$ be a Bernoulli variable for an assessment factor k for improvement option u_i, and let L be the total number of assessment factors. We define the expected value for X_{u_i} as

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

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 k, a higher value of $E[Z_k^{(u_i)}]$ 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 overhead. Thus we compute the initial belief value for u_5 as $P(X_{u_5} = 1) = 0.5\big(E[Z_1^{(u_5)}] + 0.5 * E[Z_2^{(u_5)}]\big)$ 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

To validate our models (introduced in Sections 4 and 5), we conduct four experiments on an actual corporate network

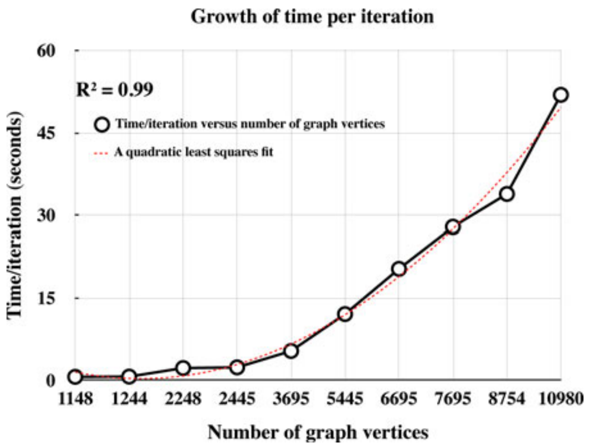


Fig. 6. The x-axis captures the number of vertices for each experiment, while the y-axis shows the average time measured in seconds to execute one iteration for Procedure 2. On average 87.99 percent of time is spent on computing the Taylor expansion.

(depicted in Fig. 5). Our experiments focus on (i) computing the ECSA values for the network, (ii) assessing security defense strategies, (iii) adding mobile device data to the analysis, and (iv) security improvement without installation of new devices.

We implemented a tool for our computational procedures (Section 4.3) in Java (with approximately 3,500 lines of code). We use (GNU Linear Programming Kit) GLPK [35], a well known open source linear programming API for our SLP-based procedure.

Our tool parses an attack graph input file (obtained from MulVAL [31]), computes the ECSA values according to various parameters, and performs security improvement analysis based on a set of improvement options and constraints.

In Fig. 6, we demonstrate the performance of our implementation. For each graph, we repeat the corresponding experiment to measure the time to compute the final expected chance of a successful attack at the graph's root vertex.

We compute ECSA values for the target graphs using our tool. We run our tool as a single threaded program on a machine with a 2.4 GHz Intel Core i7 processor and a 8 GB DDR3 memory. All our experiments converged with at most 20 iterations towards the solution. On average, 87.99 percent of the execution time for Procedure 2 is spent on the Taylor expansion from which on average 78.27 percent of the execution time is spent on symbolic differentiation performed using DJep[1] Java library for symbolic operations. The Taylor expansion is parallelizable, and scales with the number of vertices, hence can be done efficiently offline.

7.1 Experimental Setup

The target network of Fig. 5 is open to a large number of users and contains several servers and workstations. This network has low usage restrictions and allows untrusted mobile devices to enter the network without mandatory security scanning (some of the data is sanitized while

1. DJep is available on http://www.singsurf.org/djep/.

TABLE 1
Attack Graph A Is Generated with No Mobile Devices in the Network and Attack Graph B Is Generated with Two Mobile Devices

Attack Graph	Hosts	Nodes	Edges	Placement Options	Min. Size of Initial Belief Set
A: No mobile	13	483	663	206	22
B: With mobile	13	549	757	235	22

Placement options refers to the number of nodes that can be considered for the addition of an improvement node.

preserving the general structure and vulnerability information.). In this network, a connected user can easily obtain information about the network topology, and perform port scanning and operating system finger printing.

Generating the attack graphs. We used network scanning tools (such as nmap), online vulnerability repositories, and information provided by system administrators to create a network topology (depicted in Fig. 5) and the attack graphs that represent the real network. We performed wireless network scanning to confirm the connectivity of wireless devices to the network. We generate two attack graphs (Table 1) with slight variations. Attack graph A (483 nodes) assumes no mobile devices in the network (i.e., availability of mobile devices is 0 percent), while attack graph B (549 nodes) includes attack scenarios from untrusted mobile devices.

Assigning the initial beliefs. A subset of the initial belief values (22 values for attack graphs A and B) for computing ECSA values for our example attack graphs is given in Fig. 7. All the entries in the file correspond to known vulnerabilities for which a common vulnerability scoring system (CVSS) score is available. A CVSS score is a number in the range $[0, 1]$ that represents the exploitability level of a vulnerability. We use this number as an approximation of the probability of success for known vulnerabilities.

Initial belief values are required for every ground fact node. In our network, attack graph A contains 229 ground fact nodes. We use an automated technique to determine the initial belief values for all the nodes without a CVSS score. Of the 229 ground fact nodes, 161 nodes describe host access control information between two machines in the network. We assume that all the actual connections are highly reliable, thus setting an initial belief value for availability of these hosts to 0.9. Our tool automatically detects host access control nodes and sets the initial belief values for them.

Vulnerability fact node $\boldsymbol{u_i}$	$E[X_{u_i}]$
`CVE_2006_1516`	5.0
`CVE_2006_1518`	6.5
`CVE_2008_1483`	6.9
`CVE_2006_5752`	4.3
`CVE_2011_1929`	5.0
`CVE_2011_1968`	7.1
`CVE_2004_0331`	5.0
`CVE_2009_4565`	7.5
`CVE_2005_2090`	4.3
`CVE_2010_1899`	4.3

Fig. 7. First 10 entries in initial belief values file (containing 22 entries) for attack graphs A and B. We use the common vulnerability scoring system values as approximation of $E[X_{u_i}]$ for documented vulnerabilities.

The other fact nodes are in three categories: 37 nodes describe network services (such as Apache), 30 describe a vulnerability (for which we use the CVSS scores, as depicted in Fig. 7), and one node represents the existence of an attacker. Similar to the host access control nodes, we apply unified initial belief values for each category. Note that these parameters may be adjusted to test various scenarios, for example, under low probability of existence of an attacker.

7.2 Chances of a Successful Attack

Given complicated attack structures represented by an attack graph of the network, it is particularly interesting to analyze the attack to understand the weakest points of the network that enable the ultimate attack goal. In addition to computing the highest expected chance of success given an ultimate attack goal $E[X_{\mathcal{G}}]$, the solution to Equation (5) as described in Section 4.2 finds the expected chance of successful attack on intermediate attack goals that are *necessary* to achieve $\mathcal{G}$.

To verify the solution computed by our tool, consider the partial view of the attack graph A in Fig. 8. We highlight two attack vectors leading to privilege escalation on the database server, namely through compromising servers 3 and 4 (See Fig. 5). Computing the ECSA for all the nodes in

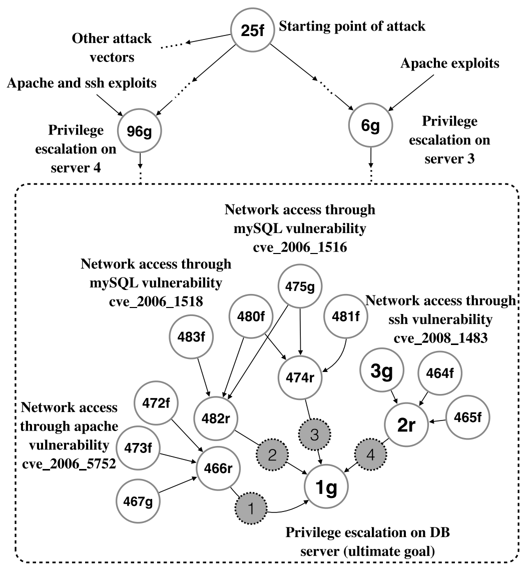

25f
Starting point of attack
Other attack vectors
Apache and ssh exploits
Apache exploits
96g
Privilege escalation on server 4
6g
Privilege escalation on server 3
Network access through mySQL vulnerability cve_2006_1516
Network access through mySQL vulnerability cve_2006_1518
475g
480f
481f
Network access through ssh vulnerability cve_2008_1483
483f
474r
3g
464f
472f
482r
3
2r
465f
Network access through apache vulnerability cve_2006_5752
473f
2
4
466r
1g
467g
1
Privilege escalation on DB server (ultimate goal)

Fig. 8. A simplified partial view of attack graph A in which nodes are numbered and labeled with node type. Nodes are labeled with an ID followed by the node type (g for goal, r for rule, and f for fact). The attack sequence starts at node 25 and proceeds with a number of alternatives among which is compromising either server 3 or server 4 through nodes 6 and 96 respectively. Both nodes 6 and 96 are predecessors of the final attack goal, that is node 1, which refers to privilege escalation on the database server.

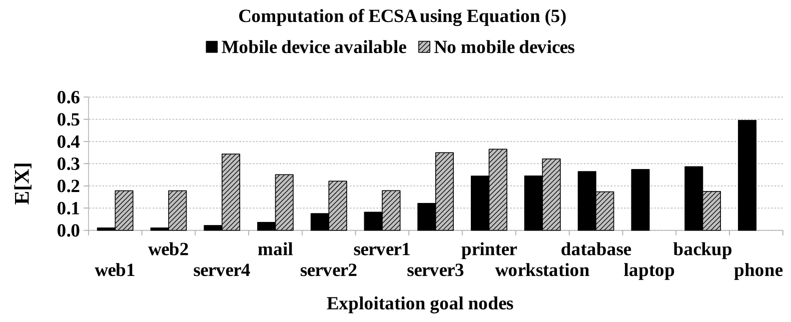


Fig. 9. ECSA values for attack graphs A (no mobile devices) and B (with mobile devices). In the experiment with mobile devices, the availability of a mobile device is captured with a random variable and is not assumed to be fixed.

the graph, the results suggest that both application servers 3 and 4 (denoted server3 and server4 in Fig. 9) 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 with a number of open ports that increase the attack surface, and thus are relatively more exposed to the outside world. The chance of successful attack on the target database server is the lowest, which is due to a better network configuration to protect it. In our target network, the probability of success (based on our computation) for compromising the database server is a function of both success probability at preceding goals (and thus taking into account the dependency on previous stages of the attack) as well as the independent probability of success at the database server itself. Our computation is more rigorous than simply regarding the probability of success at the database server to be the same as the probability for compromising the preceding servers in the attack chain.

The results obtained from this experiment significantly improve the manual inspection of network vulnerabilities even with the assistance of plain attack graphs. In addition, the ECSA values and the mathematical programming structure of Equation (5) lay the foundation for an efficient assessment of security improvement options as discussed in Section 7.3.

Fig. 9 also shows the results for the ECSA computed based on attack graph B, which are discussed in Section 7.4.

7.3 Optimal Placement of Security Products

We used the results from the previous section to find the best placement of an improvement option for the network of Fig. 5. Our improvement option is the installation of an intrusion prevention system 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 $E[X_\tau] = 0.3$.

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; note that one can choose fewer rule nodes for solving the optimal placement problem, depending on possible placement constraints.). Then we modify the original attack graph to include improvement fact nodes as predecessors to each $u \in N_{ra}$.

Cursory 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 $E[X_\mathcal{G}]$. The 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 on $\mathcal{G}$ (i.e., the database server) in the improved attack graph and the original attack graph, respectively.

The results in 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 being used to attack 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 by the 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 (obtained during the course of solving the minimization problem (7)) suggests 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.

TABLE 2
Optimal Selection for IPS Installation for Attack Graphs A and B

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

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.

TABLE 3
Optimal Selection for Closing a Single Port with the Best Effect on the Security of the Network

Rank	Attack Graph	Machine, Port	$E'[X_{\mathcal{G}}]$	$E[X_{\mathcal{G}}]$	% ↓
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

7.4 Effect of Mobile Devices

The network architecture presented in Fig. 5 is also vulnerable to threats from mobile devices. For example, in the network of Fig. 5, 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 values for 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). Therefore, the results of the experiment on attack graph B (Fig. 9) show lower values for exploiting the application servers, but higher values for exploiting the smartphone and the laptop (with highly scored known software vulnerabilities), the workstation, and the printer. This is because the mobile devices in the network of Fig. 5 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 a successful 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 in exploiting the database server is increased by 52.44 percent.

7.5 Improving Network Configuration

Our optimal recommendation method is capable of computing 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 $E[X_{\mathcal{G}}]$ (the optimum value for (5)) 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.

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 yielding 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.

All the analyses (including vulnerability and optimization analyses) conducted in our experiment finished within several seconds for the attack graphs used.

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 the ability to quantitatively analyze the chance of successful attack in the presence of uncertainties about the configuration of a dynamic network and routes of potential attacks.

For 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. For instance, an important issue is noise elimination in the initial belief set of values. This is an important problem that if solved will lead to the production of more accurate results.

Acknowledgments

This work was supported in part by Kuwait University Research Grant No. [ZQ02/14]. Danfeng (Daphne) Yao is the corresponding author.

References

[1] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proc. Comput. Security Appl. Conf.*, 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. Norwell, MA, USA: Kluwer, 2003, ch. 5.
[3] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proc. 15th IEEE Comput. Security Found. Workshop*, 2002, pp. 49–63.
[4] S. Fenz, "An ontology-and Bayesian-based approach for determining threat probabilities," in *Proc. 6th ACM Symp. Inform., Comput. Commun. Security*, 2011, pp. 344–354.
[5] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic Bayesian network," in *Proc. 4th ACM Workshop Quality Protection*, 2008, pp. 23–30.
[6] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Trans. Dependable Secure Comput.*, 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 *Proc. 40th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2010, pp. 211–220.
[8] S. Noel, S. Jajodia, L. Wang, and A. Singhal, "Measuring security risk of networks using attack graphs," *Int. J. Next-Generation Comput.*, vol. 1, no. 1, pp. 1–11, Jul. 2010.
[9] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Proc. 22nd Annu. IFIP WG 11.3 Working Conf. Data Appl. Security*, 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*, vol. 4219. New York, NY, USA: Springer Berlin, 2006, pp. 127–144.
[11] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *Proc. 13th Eur. Symp. Res. Computer Security: Comput. Security*, 2008, pp. 18–34.

[12] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *Proc. 42nd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 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 *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 204–213.

[14] S. Noel and S. Jajodia, "Optimal IDS sensor placement and alert prioritization using attack graphs," *J. Netw. Syst. Manage.*, vol. 16, no. 3, pp. 259–275, Sep. 2008.

[15] C. Phillips and L. P. Swiler, "A Graph-based System for Network-vulnerability Analysis," in *Proc. Workshop New Security Paradigms*, 1998, pp. 71–79.

[16] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. (2002). Automated generation and analysis of attack graphs, in *Proc. IEEE Symp. Security Privacy* [Online]. Available: http://portal.acm.org/citation.cfm?id=829514.830526

[17] Y. Dou, K. Zeng, Y. Yang, and D. Yao, "MadeCR: Correlation-based Malware detection for cognitive radio," in *Proc. IEEE Conf. Comput. Commun.*, 2015.

[18] M. Albanese, S. Jajodia, A. Pugliese, and V. Subrahmanian, ''Scalable analysis of attack scenarios," in *Proc. 16th Eur. Conf. Res. Comput. Security*, 2011, pp. 416–433.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Stanford Univ., Stanford, CA, USA, Tech. Rep. 1999-66, Sept. 1999.

[20] Z. Zhang, F. Naït-Abdesselam, X. Lin, and P.-H. Ho, "A Model-based Semi-quantitative Approach for Evaluating Security of Enterprise Networks," in *Proc. ACM Symp. Appl. Comput.*, 2008, pp. 1069–1074.

[21] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A weakest-adversary security metric for network configuration security analysis," in *Proc. 2nd ACM Workshop Quality Protection*, 2006, pp. 31–38.

[22] L. Wang, A. Singhal, and S. Jajodia. (2007). Measuring the overall security of network configurations using attack graphs, in *Proc. 21st Annu. IFIP WG 11.3 Working Conf. Data Appl. Security*, pp. 98–112 [Online]. Available: http://dl.acm.org/citation.cfm?id=1770560.1770573

[23] S. Bhatt, W. Horne, and P. Rao, "On computing enterprise IT risk metrics," in *Proc. 26th IFIP TC 11 Int. Inf. Security Conf. Future Challenges Security Privacy Acad. Ind.*, 2011, vol. 354, pp. 271–280.

[24] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Comput. Commun.*, vol. 29, no. 18, pp. 3812–3824, Nov. 2006.

[25] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah, "Distilling critical attack graph surface iteratively through minimum-cost SAT solving," in *Proc. 27th Annu. Comput. Security Appl. Conf.*, 2011, pp. 31–40.

[26] G. Lee, I.-s. Ko, and T.-H. Kim, "A vulnerability assessment tool based on OVAL in system block model," in *Proc. Int. Conf. Intell. Comput.*, 2006, pp. 1115–1120.

[27] X. Ou, S. Govindavajhala, and A. W. Appel. (2005). Mulval: A logic-based network security analyzer, in *Proc. 14th Conf. USENIX Security Symp.*, pp. 113–128 [Online]. Available: http://portal.acm.org/citation.cfm?id=1251398.1251406

[28] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, pp. 385–463, May 2004.

[29] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming*. New York, NY, USA: Wiley, 2005.

[30] F. Palacios-Gomez, L. Lasdon, and M. Engquist. (1982). Nonlinear optimization by successive linear programming. *Manage. Sci.* [Online]. *28(10)*, pp. 1106–1120. Available: http://www.jstor.org/stable/2630940

[31] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 336–345.

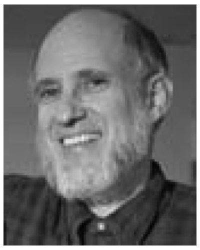
[32] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Trans. Dependable Sec. Comput.*, vol. 11, no. 1, pp. 30–44, Jan./Feb. 2014.

[33] D. Balzarotti, M. Monga, and S. Sicari, "Assessing the risk of using vulnerable components," in *Quality of Protection*, series Advances in Information Security, vol. 23, D. Gollmann, F. Massacci, and A. Yautsiukhin, Eds. New York, NY, USA: Springer, 2006, pp. 65–77.

[34] M. McQueen, T. McQueen, W. Boyer, and M. Chaffin, "Empirical estimates and observations of 0-day vulnerabilities," in *Proc. 42nd Hawaii Int. Conf. Syst. Sci.*, Jan. 2009, pp. 1–12.

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

Hussain M.J. Almohri received the BS degree from Kuwait University and the MS degree from Kansas State University. In 2013, he received the PhD degree from Virginia Tech. He started as an assistant professor of computer science at Kuwait University. His research focuses on mobile system security and quantitative analysis of security. He has served as a reviewer for several IEEE journals. He is a member of the ACM, IEEE, and USENIX.



Layne T. Watson received the BA degree (magna cum laude) in psychology and mathematics from the University of Evansville, Indiana, in 1969, and the PhD degree in mathematics from the University of Michigan, Ann Arbor, in 1974. He is a professor of computer science, mathematics, and aerospace and ocean engineering at Virginia Polytechnic Institute and State University. His professional service includes stints as associate editor of *ORSA Journal on Computing*, *SIAM Journal on Optimization*, *Computational Optimization and Applications*, *Evolutionary Optimization*, *Engineering Computations*, and the *International Journal of High Performance Computing Applications*, and is a senior editor of *Applied Mathematics and Computation*. He has published well more than 300 refereed journal articles and 200 refereed conference papers. He is a fellow of the IEEE, the National Institute of Aerospace, and the International Society of Intelligent Biological Medicine.

Danfeng (Daphne) Yao received the PhD degree in computer science from Brown University in 2007. She is an associate professor and L-3 Faculty fellow in the Department of Computer Science at Virginia Tech. She received the US National Science Foundation (NSF) CAREER Award in 2010 for her work on human-behavior driven malware detection, and most recently ARO Young Investigator Award for her semantic reasoning for mission-oriented security work in 2014. She received the Outstanding New Assistant Professor Award from Virginia Tech College of Engineering in 2012. She has several Best Paper Awards (ICICS '06, CollaborateCom '09, and ICNP '12). She is a senior member of the IEEE.

Xinming "Simon" Ou received the PhD degree in computer science from Princeton University. He is an associate professor of computer science and the Peggy and Gary Edwards Chair in Engineering at Kansas State University. His research interests include designing better technologies to facilitate cyberdefense. He is a 2010 National Science Foundation Faculty Early Career Development award recipient and three-time winner of HP Labs Innovation Research Program Award. He is a member of the IEEE.