Dynamic Security Risk Management Using Bayesian Attack Graphs

Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray, *Member, IEEE*

Abstract—Security risk assessment and mitigation are two vital processes that need to be executed to maintain a productive IT infrastructure. On one hand, models such as attack graphs and attack trees have been proposed to assess the cause-consequence relationships between various network states, while on the other hand, different decision problems have been explored to identify the minimum-cost hardening measures. However, these risk models do not help reason about the causal dependencies between network states. Further, the optimization formulations ignore the issue of resource availability while analyzing a risk model. In this paper, we propose a risk management framework using Bayesian networks that enable a system administrator to quantify the chances of network compromise at various levels. We show how to use this information to develop a security mitigation and management plan. In contrast to other similar models, this risk model lends itself to dynamic analysis during the deployed phase of the network. A multi-objective optimization platform provides the administrator with all trade-off information required to make decisions in a resource constrained environment.

Index Terms—security risk assessment, mitigation analysis, bayesian belief networks, attack graph.

1 Introduction

TRADITIONAL information security planning and management begins with risk assessment that determines threats to critical resources and the corresponding loss expectancy. A number of researchers have proposed risk assessment methods by building security models of network systems, using paradigms like attack graphs [1], [2], [3], [4], [5] and attack trees [6], [7], [8], [9], and then finding attack paths in these models to determine scenarios that could lead to damage. However, a majority of these models fail to consider the attacker's capabilities and, consequently, the likelihood of a particular attack being executed. Without these considerations, threats and their impact can be easily misjudged.

To alleviate such drawbacks, Dantu et al. [10] propose a probabilistic model to assess network risks. They model network vulnerabilities using attack graphs and apply Bayesian logic to perform risk analysis. Liu and Man [11] use Bayesian networks to model potential attack paths in a system, and develop algorithms to compute an optimal subset of attack paths based on background knowledge of attackers and attack mechanisms. In both Dantu et al. and Liu and Man's works, nodes in the attack graph are assigned a probability value that describes the likelihood of attack on a node. They compute the likelihood of system compromise by chaining Bayesian belief rules on top of the assigned probabilities. The organizational risk is then computed as the product of the likelihood of system compromise and the value of expected loss. A limitation with both these works is that none of them specify how the conditional probability value of an attack on each node is computed. Further, they do not address the problem of optimal risk management.

System administrators are often interested in assessing the risk to their systems and determining the best possible way to defend their network in terms of an enumerated set of hardening options. Risk assessment methods such as those discussed earlier have been adopted by researchers to determine a set of potential safeguards, and related security control installation costs. Noel et al. use exploit dependency graphs to compute minimum-cost hardening measures [12]. Jha et al. [2] determine the minimal set of attacks critical for reaching a goal and then find the minimal set of security measures that cover this set of attacks.

While such cost analysis techniques are useful, they miss out on one major issue. The system administrator often has to work within a given set of budget constraints that may preclude her from implementing all possible hardening measures or even measures that cover all the weak spots. Thus, the system administrator needs to find a trade-off between the cost of implementing a subset of security hardening measures (from a set of measures that can potentially close all attack paths) and the damage that can be potentially inflicted on the system after the security decision has been made (the residual damage).

Dewri et al. [13] first formulated this problem (the so-called "system administrators' dilemma") as a series of multi-objective optimization problems. The solutions to these problems allow one to select a subset of hardening measures so that the total cost of implementing them is not only minimized but also within a fixed budget and, at the same time, the residual damage is minimized. One of the significant contributions of Dewri et al.'s work is the development of an attack tree model of network risks that drives the solution methodology. The attack tree model is able to better guide the optimization process by providing knowledge about the attributes that make an attack possible. While this work makes significant contribution towards appreciating the security planning process as something

N. Poolsappasit (nayot@mst.edu) is with the Missouri University of Science and Technology.
R. Dewri (rdewri@cs.du.edu) is with the University of Denver.
I. Ray (indrajit@cs.colostate.edu) is with the Colorado State University.

beyond simple risk assessment, it has one significant shortcoming. The authors' modeling of the problem is a static one. There is, however, a dynamic aspect to the security planning process. For every attack, there is a probability of occurrence that can change during the life time of a system depending on what the contributing factors for the attack are and how they are changing. During run time, the system administrator may need to revise her decision based on such emerging security conditions. Dewri et al.'s attack tree model does not allow such dynamic security planning.

To address these limitations, the current work makes five major contributions.

- We propose an alternative method of security risk assessment that we call *Bayesian Attack Graph*s (BAGs). In particular, we adapt the notion of Bayesian belief networks so as to encode the contribution of different security conditions during system compromise. Our model incorporates the usual cause-consequence relationships between different network states (as in attack graphs and attack trees) and, in addition, takes into account the likelihoods of exploiting such relationships.
- We propose a method to estimate an organization's security risk from different vulnerability exploitations based on the metrics defined in the Common Vulnerability Scoring System (CVSS) [14].
- We develop a model to quantify the expected return on investment based on a user specified cost model and likelihoods of system compromise.
- We model the risk mitigation stage as a discrete reasoning problem and propose a genetic algorithm to solve it. The algorithm can identify optimal mitigation plans in the context of both single and multi-objective analysis.
- Last, but not the least, we discuss how the above contributions collectively provide a platform for static and dynamic analysis of risks in networked systems.

The rest of the paper is organized as follows. The test network used to illustrate our problem formulation and solution is described in Section 2. Section 3 presents the formalism for a Bayesian Attack Graph model. The likelihood estimation method in static and dynamic scenarios is discussed in Section 4. The risk mitigation process along with the expected cost computations is presented in Section 5. Results and discussion are presented in Section 6 followed by a discussion of related works in Section 7. Finally, we conclude the paper in Section 8.

2 A Test Network

Fig. 1 depicts the test network used in this study. The network consists of eight hosts located within two subnets. A DMZ tri-homed firewall is installed with preset policies to ensure that the Web server, Mail server and the DNS server, located in the DMZ network, are separated from the local network. The firewall has a strong set of policies (shown in the inset table) to prevent remote access to the internal hosts. In particular, all machines in the DMZ zone passively receive service requests and only respond to the sender as needed. However, in order to accommodate Web service's transactions, the Web server is allowed to send SQL queries to the SQL server located in the trusted zone on a designated channel. Local machines are located behind a NAT firewall so that all communications to external parties are delivered through the Gateway server. In addition, all local desktops, including the administrator machine, have remote desktop enabled to facilitate remote operations for company employees working from remote sites. The remote connections are monitored by SSHD installed in the Gateway server.

A list of initial vulnerabilities in this test network is listed in Table 1. These vulnerabilities can produce more than 20 attack scenarios with different outcomes, ranging from information leakage to system compromise. Moreover, two of these scenarios use machines in the DMZ zone to compromise a local machine in the trusted zone.

3 Modeling Network Attacks

We extend the notion of Bayesian networks as presented by Liu and Man [11] to encode the contributions of different security conditions during a system compromise. We term such a Bayesian network a *Bayesian Attack Graph* (BAG).

Different properties of the network effectuate different ways for an attacker to compromise a system. We first define an *attribute-template* that allows us to categorize these network properties for further analysis.

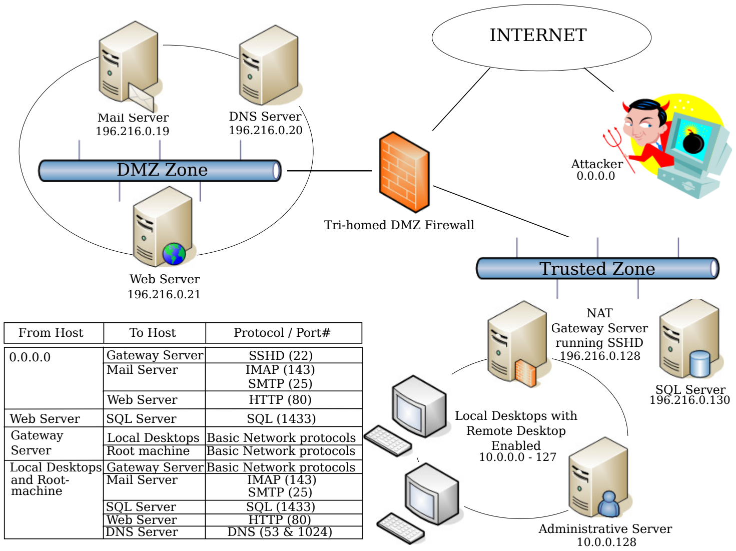
Definition 1: Attribute-Template: An attribute-template is a generic property of the network that includes, but not limited to, the following:

1) system vulnerabilities (as reported in vulnerability databases such as BugTraq, CERT/CC, or netcat),
2) (insecure) system properties such as unsafe security policy, corrupted file/memory access permission, or read-write access in file structure,
3) (insecure) network properties such as unsafe network condition, unsafe firewall properties, unsafe device/peripheral access permission, and
4) access privilege such as user account, guest account, or root account.

An attribute-template helps us categorize the properties of the network that may be useful to an attacker. For example, *"SSH buffer overflow vulnerability in FTP server"* can be considered as an instance of the system vulnerabilities template. Similarly, *"user access on local machine"* is an instance of the access privilege template. Such templates let us specify the properties as random variables. We define an *attribute* with such a concept in mind.

Definition 2: Attribute: An attribute is a Bernoulli random variable representing the state of an instance of an attribute-template.

An attribute S is thus associated with a state – *True* ($S = 1/T$) or *False* ($S = 0/F$) – and a probability $Pr(S)$. The state signifies the truth value of the proposition underlying the instance of the attribute template. For example, the instance S :"*user access on Local machine*" is an attribute when associated with a truth value signifying whether an



From Host	To Host	Protocol / Port#
0.0.0.0	Gateway Server	SSHD (22)
	Mail Server	IMAP (143) SMTP (25)
	Web Server	HTTP (80)
Web Server	SQL Server	SQL (1433)
Gateway Server	Local Desktops	Basic Network protocols
	Root machine	Basic Network protocols
Local Desktops and Root-machine	Gateway Server	Basic Network protocols
	Mail Server	IMAP (143) SMTP (25)
	SQL Server	SQL (1433)
	Web Server	HTTP (80)
	DNS Server	DNS (53 & 1024)

Fig. 1. Test-bed network model.

TABLE 1
INITIAL LIST OF VULNERABILITIES IN TEST NETWORK.

Host	Vulnerability	CVE#	Type of Attack
Local desktops (10.0.0.0-127)	Remote login LICQ Buffer Overflow (BOF) MS Video ActiveX Stack BOF	CA 1996-83 CVE 2001-0439 CVE 2009-0015	remote-2-user remote-2-user remote-2-root
Admin machine (10.0.0.128)	MS SMV service Stack BOF	CVE 2008-4050	local-2-root
Gateway server (196.216.0.128)	OpenSSL uses predictable random Heap corruption in OpenSSH Improper cookies handler in OpenSSH	CVE 2008-0166 CVE 2003-0693 CVE 2007-4752	information leakage local-2-root authentication bypass
SQL Server (196.216.0.130)	SQL Injection	CVE 2008-5416	remote-2-root
Mail Server (196.216.0.19)	Remote code execution in SMTP Error message information leakage Squid port scan vulnerability	CVE 2004-0840 CVE 2008-3060 CVE 2001-1030	remote-2-root account information theft information leakage
DNS Server (196.216.0.20)	DNS Cache Poisoning	CVE 2008-1447	integrity
Web Server (196.216.0.21)	IIS vulnerability in WebDAV service	CVE 2009-1535	remote-2-local authentication bypass

attacker has user access on the local machine. We shall also use the term "compromised" to indicate the *true* (or $S = 1$) state of an attribute. Further, $Pr(S)$ is the probability of the attribute being in state $S = 1$. Consequently, $Pr(\neg S) = 1 - Pr(S)$ is the probability of the state being $S = 0$. The success or failure of an attacker reaching its goal depends mostly on the states of the attributes in a network. It also lays the foundations for a security manager to analyze the effects of forcing some attributes to the *false* state using security measures. We formally define a Bayesian Attack Graph (BAG) to capture the cause-consequence relationships between such attributes.

Definition 3: ATOMIC ATTACK: Let S be a set of attributes. We define $\mathcal{A}$, a conditional dependency between a pair of attributes, as a mapping $\mathcal{A} : S \times S \to [0,1]$. Then, given $S_{pre}, S_{post} \in S$, $a : S_{pre} \mapsto S_{post}$ is called an atomic attack if

1) $S_{pre} \neq S_{post}$,
2) given $S_{pre} = 1$, $S_{post} = 1$ with probability $\mathcal{A}(S_{pre}, S_{post}) > 0$, and
3) $\nexists S_1, \dots, S_j \in S - \{S_{pre}, S_{post}\}$ such that $\mathcal{A}(S_{pre}, S_1) > 0, \mathcal{A}(S_1, S_2) > 0, \dots,$ and $\mathcal{A}(S_j, S_{post}) > 0$.

An atomic attack allows an attacker to compromise the attribute S_{post} from S_{pre} with a non-zero probability of success. Although, given a compromised attribute, another attribute can be compromised with positive probability using a chain of other attributes, the third condition in the definition does not allow such instances to be considered as atomic attacks. Instead, each step in such a chain is an atomic attack. Informally, an attack is associated with a vulnerability exploitation, denoted by e_i, which takes the attacker from one network state (S_{pre}) to another (S_{post}). The probability of an exploitation, $Pr(e_i)$, states the ease with which an attacker can perform the exploitation. Hence, we say that $\mathcal{A}(S_{pre}, S_{post}) = Pr(e_i)$, and S_{pre} and S_{post} are respectively called a *precondition* and *postcondition* of the attack a, denoted by *pre(a)* and *post(a)* respectively.

An attack relates the states of two different attributes so as to embed a cause-consequence relationship between the two. For example, for the attributes S_{pre} ="*sshd BOF vulnerability on machine* A" and S_{post} ="*root access privilege on machine* A", the attack $S_{pre} \mapsto S_{post}$ is associated with the e_i ="*sshd buffer overflow*" exploit. Using this exploit, an attacker can achieve root privilege on a machine, provided the machine has the sshd BOF vulnerability. $\mathcal{A}(S_{pre}, S_{post})$ is the probability of success of the exploit, i.e. $\mathcal{A}(S_{pre}, S_{post}) = Pr(e_i)$.

Definition 4: Bayesian Attack Graph: Let S be a set of attributes and A be the set of atomic attacks defined on S. A Bayesian Attack Graph is a tuple $BAG = (S, \tau, \varepsilon, \mathcal{P})$, where

1) $S = N_{internal} \cup N_{external} \cup N_{terminal}$. $N_{external}$ denotes the set of attributes S_i for which $\nexists a \in A | S_i = post(a)$. $N_{internal}$ denotes the set of attributes S_j for which $\exists a_1, a_2 \in A | [S_j = pre(a_1)$ and $S_j = post(a_2)]$. $N_{terminal}$ denotes the set of attributes S_k for which $\nexists a \in A | S_k = pre(a)$.
2) $\tau \subseteq S \times S$. An ordered pair $(S_{pre}, S_{post}) \in \tau$ if $S_{pre} \mapsto S_{post} \in A$. Further, for $S_i \in S$, the set $Pa[S_i] = \{S_j \in S | (S_j, S_i) \in \tau\}$ is called the *parent set* of S_i.
3) ε is a set of decomposition tuples of the form $\langle S_j, d_j \rangle$ defined for all $S_j \in N_{internal} \cup N_{terminal}$ and $d_j \in \{AND, OR\}$. d_j is AND if $S_j = 1 \Rightarrow \forall S_i \in Pa[S_j], S_i = 1$. d_j is OR if $S_j = 1 \Rightarrow \exists S_i \in Pa[S_j], S_i = 1$.
4) $\mathcal{P}$ is a set of discrete conditional probability distribution functions. Each attribute $S_j \in N_{internal} \cup N_{terminal}$ has a discrete local conditional probability distribution (LCPD) representing the values of $Pr(S_j \mid Pa[S_j])$.

Fig. 2 shows the BAG for our test network. We have developed an in-house tool to generate such BAGs. The tool takes as input an initial vulnerability table, generated by a vulnerability scanner, and the network topology. Using a sequence of SQL queries on a vulnerability exposure database, the tool creates consequence attributes for the graph until no further implications can be derived.

The BAG in Fig. 2 depicts a clear picture of 20 different attack scenarios. Each node is a Bernoulli random variable (S_i) representing the state variable of the attribute. The set $N_{external}$ represents the entry points of the graph. These nodes reflect an attacker's capability as discovered in a threat-source model. $N_{terminal}$ resemble the end points in the graph. These nodes reflect casualty at the end of each attack scenario. The set of ordered pair, τ, reflects the edges in the graph. The existence of an edge between two nodes imply that there is a causal dependency between their states, signified by the decomposition at each node. AND-decomposition signifies that the compromised state of a node implies that all nodes in its parent set have also been compromised. Similarly, OR-decomposition signifies that at least one parent node is in the *true* state. Note that these decompositions are uni-directional. For instance, under AND-decomposition, compromising all nodes in the parent set does not necessarily imply the node itself has been compromised. This is because the attacks relating the node with its parents can have varying levels of difficulty, or in other words, different probabilities of success. Hence, although the preconditions of the attacks have been met, there can still be a non-zero probability that the attacker is unable to carry out all the exploits successfully. The existence of this probability is what primarily differentiates a BAG from a classical attack graph. The probabilities are captured in the local conditional probability distribution of the node. The LCPD is a set of probability values specifying the chances of the node being compromised, given different combination of states of its parents.

Definition 5: Local Conditional Probability Distribution: Let $BAG = (S, \tau, \varepsilon, \mathcal{P})$ be a BAG and $S_j \in N_{internal} \cup N_{terminal}$. For $S_i \in Pa[S_j]$, let e_i be the vulnerability exploitation associated with the attack $S_i \mapsto S_j$. A local conditional probability distribution (LCPD) function of S_j, mathematically equivalent to $Pr(S_j \mid Pa[S_j])$, is defined as follows.

1) $d_j = AND$

$$Pr(S_j \mid Pa[S_j]) = \begin{cases} 0, & \exists S_i \in Pa[S_j] \mid S_i = 0 \\ Pr(\bigcap_{S_i=1} e_i), & otherwise \end{cases}$$

2) $d_j = OR$

$$Pr(S_j \mid Pa[S_j]) = \begin{cases} 0, & \forall S_i \in Pa[S_j], S_i = 0 \\ Pr(\bigcup_{S_i=1} e_i), & otherwise \end{cases}$$

To compute the local conditional probabilities when multiple exploits are involved, we proceed as follows. For AND-decomposition, each vulnerability exploitation is a distinct event. The chance of compromising the target node depends on the success of each individual exploit. Therefore, we use the product rule in probability to derive $Pr(\bigcap_{S_i=1} e_i)$ as

$$Pr(\bigcap_{S_i=1} e_i) = \prod_{S_i=1} Pr(e_i). \tag{3.1}$$

For OR-decomposition, Liu et al. observed that the joint probability is equivalent to the noisy-OR operator [11], given as

$$Pr(\bigcup_{S_i=1} e_i) = 1 - \prod_{S_i=1} [1 - Pr(e_i)]. \tag{3.2}$$

4 Security Risk Assessment with BAG

Security risk management consists of threat analysis, risk assessment, loss expectancy, potential safeguards and risk mitigation analysis. Using a BAG, the administrator performs risk assessment and risk mitigation as follows.

1) *Static Risk Assessment*: Risk assessment begins with the identification of system characteristics, potential threat sources and attacker capabilities. Threat sources are represented as the external nodes in a BAG, along with their impact on other network attributes. One set of attributes act as preconditions to an exploit, which when successfully executed by an attacker, can make the network state favorable for subsequent exploits. Estimating the amount of risk at each node therefore requires some judgment on attacker capabilities. Often this judgment is indirectly stated as the system administrator's subjective belief on the likelihood of a threat source becoming active

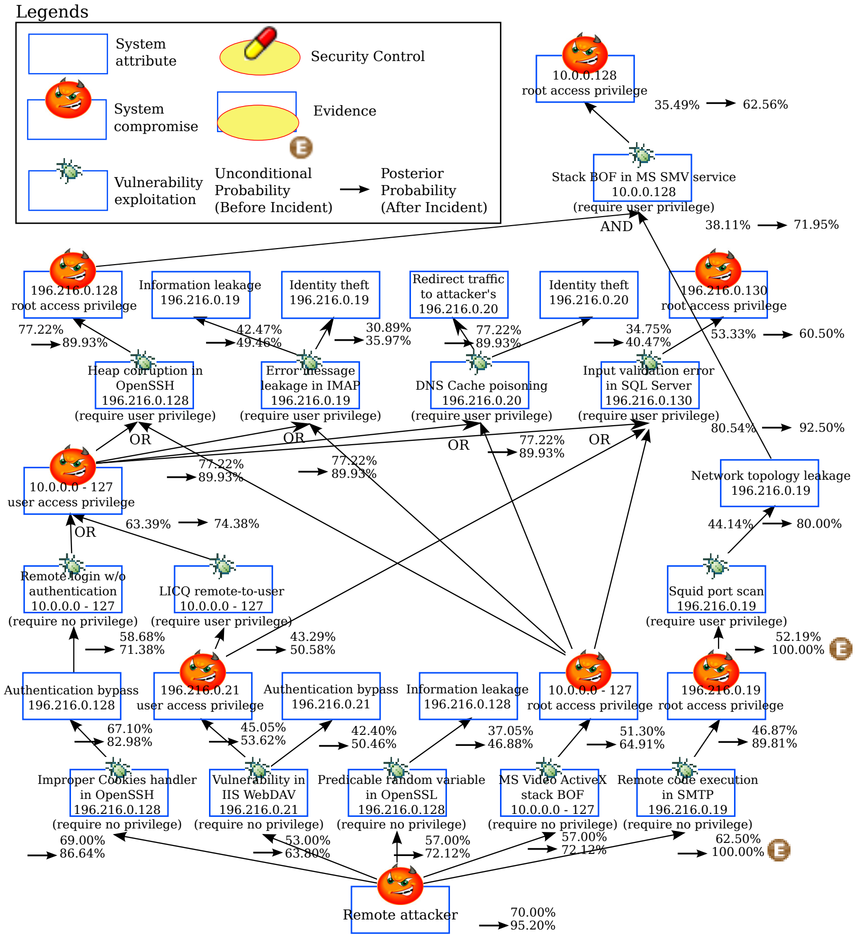

Legends
System attribute
Security Control
System compromise
Evidence
E
Vulnerability exploitation
Unconditional Probability (Before Incident)
Posterior Probability (After Incident)
10.0.0.128 root access privilege
35.49% → 62.56%
Stack BOF in MS SMV service 10.0.0.128 (require user privilege)
AND
38.11% → 71.95%
196.216.0.128 root access privilege
77.22% → 89.93%
Information leakage 196.216.0.19
42.47% → 49.46%
Identity theft 196.216.0.19
30.89% → 35.97%
Redirect traffic to attacker's 196.216.0.20
77.22% → 89.93%
Identity theft 196.216.0.20
34.75% → 40.47%
196.216.0.130 root access privilege
53.33% → 60.50%
Heap corruption in OpenSSH 196.216.0.128 (require user privilege)
Error message leakage in IMAP 196.216.0.19 (require user privilege)
DNS Cache poisoning 196.216.0.20 (require user privilege)
Input validation error in SQL Server 196.216.0.130 (require user privilege)
80.54% → 92.50%
OR
OR
OR
OR
77.22% → 89.93%
77.22% → 89.93%
77.22% → 89.93%
10.0.0.0 - 127 user access privilege
63.39% → 74.38%
OR
Network topology leakage 196.216.0.19
44.14% → 80.00%
Remote login w/o authentication 10.0.0.0 - 127 (require no privilege)
LICQ remote-to-user 10.0.0.0 - 127 (require user privilege)
Squid port scan 196.216.0.19 (require user privilege)
58.68% → 71.38%
43.29% → 50.58%
52.19% → 100.00% E
Authentication bypass 196.216.0.128
196.216.0.21 user access privilege
Authentication bypass 196.216.0.21
Information leakage 196.216.0.128
10.0.0.0 - 127 root access privilege
196.216.0.19 root access privilege
67.10% → 82.98%
45.05% → 53.62%
42.40% → 50.46%
37.05% → 46.88%
51.30% → 64.91%
46.87% → 89.81%
Improper Cookies handler in OpenSSH 196.216.0.128 (require no privilege)
Vulnerability in IIS WebDAV 196.216.0.21 (require no privilege)
Predicable random variable in OpenSSL 196.216.0.128 (require no privilege)
MS Video ActiveX stack BOF 10.0.0.0 - 127 (require no privilege)
Remote code execution in SMTP 196.216.0.19 (require no privilege)
69.00% → 86.64%
53.00% → 63.80%
57.00% → 72.12%
57.00% → 72.12%
62.50% → 100.00% E
Remote attacker
70.00% → 95.20%

Fig. 2. BAG of test network with unconditional probabilities and posterior probabilities given two attack incidences (marked by Ⓔ).

and the difficulty of an exploit. The former is represented by the probabilities $Pr(S_i)$ for all $S_i \in N_{external}$, also called the *prior probabilities*, and is subjectively assigned by the administrator. The latter is incorporated into an internal node's LCPD. Thereafter, given the prior probability values and the LCPDs, we can compute the *unconditional probability* $Pr(S_j)$ for any node $S_j \in N_{internal} \cup N_{terminal}$. These risk estimates can be used to help locate weak spots in the system design and operations.

2) *Dynamic Risk Assessment*: A deployed system may experience first hand attack incidents during its life cycle. Similar to [30], a BAG can be used for correlating alerts, hypothesizing missing and predicting future attacks. Formally, an attack incident is evidence that an attribute is in the *true* state. A security administrator may then want to investigate how these incidents impact the risk estimates initially derived solely based on subjective beliefs. Knowledge about attack incidents is therefore used to update the probabilities using the Bayesian inference techniques of forward and backward propagation. Forward propagation updates the probability on successor attributes that are directly influenced by the evidences. Backward propagation corrects/adjusts the initial hypothesis on all prior attributes. Thereafter, the *posterior probabilities* (updated unconditional probabilities) reflect the likelihoods of other potential outcomes under the light of detected events.

3) *Risk Mitigation Analysis*: Risk assessment paves the way for efficient decision making targeted at countering risks either in a proactive or reactive manner. Given a set of security measures (e.g. firewall, access control policy, cryptography, etc.), we can design the security plan which is the most resource efficient in terms of reducing risk levels in the system. This can be done before the deployment (static mitigation) or in response to attack incidents (dynamic mitigation).

4.1 Probability of vulnerability exploitation

In order to compute the local conditional probability distribution (LCPD) of an attribute, the administrator needs to estimate the probability of success while an attacker exploits a known vulnerability exploitation. We use the metrics defined in NIST's Common Vulnerability Scoring System (CVSS) [14] to estimate the attack likelihood.

A CVSS score is a decimal number on a scale of 0 to 10. It is composed of three groups – *base*, *temporal* and *environmental*. The base metrics quantify the intrinsic characteristics of a vulnerability with two sub-scores – (i) the exploitability sub-score, composed of the access vector (*B_AV*), access complexity (*B_AC*) and authentication instances (*B_AU*), and (ii) the impact sub-score, expressing the potential damage on confidentiality (*B_C*), integrity (*B_I*) and availability (*B_A*). The temporal metrics quantify dynamic aspects of a vulnerability on the environment around the organization. These metrics take into account the availability of exploitable tools and techniques (*T_E*), remediation level (*T_RL*) and report confidence (*T_RC*). The environmental metrics quantify two aspects of impact that are dependent on the environment surrounding the organization. More details on CVSS metrics and their scoring computation can be found in the CVSS guide [14]. In this study, we are interested in likelihood estimation and hence the impact sub-score and environmental metrics are ignored in the analysis. Given the vulnerability exposure information (CVSS attributes), the probability of success $Pr(e_i)$ while executing a given vulnerability exploitation e_i is computed from CVSS's Exploitability sub-score as the following.

$$Pr(e_i) = 2 \times B\_AV \times B\_AC \times B\_AU \tag{4.1}$$

4.2 Local conditional probability distributions

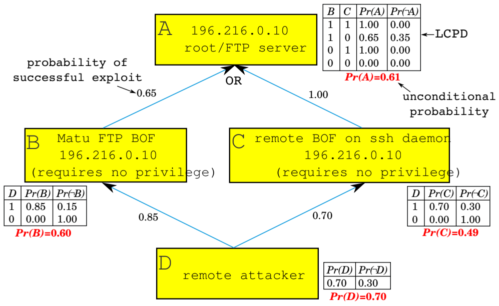
To show how various conditional probabilities are computed, we refer to a simple BAG in Fig. 3. Nodes A:*"root/FTP server"*, B:*"Matu FTP BOF"* and C:*"remote BOF on ssh"* are internal attributes, while node D:*"remote attacker"* is an external attribute. A is the successor of B and C which in turn are successors of D. The values on the edges reflect the probability of success of the associated vulnerability exploitation, computed by following the procedure described in the previous section. We begin by assigning a prior probability of $Pr(D) = 0.7$ to the external attribute D. This probability represents the administrator's subjective belief on the chances of a remote attack. For the nodes A, B and C, we calculate LCPDs by the equations previously defined in Def. 5. For example, for node A, there are 2^2 marginal cases given the two parents B and C. The decomposition at the node dictates the rule to follow while computing the local conditional probability for each case.

4.3 Unconditional probability to assess security risk

Once the LCPDs have been assigned to all attributes in the BAG, we can merge the marginal cases at each node to obtain the unconditional probability at the node. Given a set of Bernoulli random variables $S = \{S_1, \ldots, S_n\}$ in a Bayesian belief network, the joint probability of all the variables is given by the *chain rule* as

$$Pr(S_1, \ldots, S_n) = \prod_{i=1}^{n} Pr(S_i \mid Pa[S_i]). \tag{4.2}$$

It is important to note that Bayes theorem can only be applied to acyclic graphs. Cycles can often occur in attack graphs when the graph is trying to model different attack scenarios. However, in terms of value gained, a cycle does not increase the likelihood of an attack or change the outcomes of the attack. Consider the BAG in Fig. 2. Once the attacker has gained root privilege on the target machine, the attacker can execute arbitrary code, including overriding existing commands with ones that are vulnerable to BOF attacks. This can create a cycle pointing back to the root compromise state. However, there is no additional value gained by these actions. The attacking capability or exploitation probability from the defender's perspective (or even from the attacker's perspective) does not change. Our Bayesian attack graph models "why an attack can happen" rather than "how the attack can happen" [24]. Without doubt, the cycle constitutes a new attack scenario, but from the value gained perspective, such cycles can be disregarded using the monotonicity constraint [1].



B	*C*	*Pr(A)*	*Pr(¬A)*
1	1	1.00	0.00
1	0	0.65	0.35
0	1	1.00	0.00
0	0	0.00	0.00

D	*Pr(B)*	*Pr(¬B)*
1	0.85	0.15
0	0.00	1.00

D	*Pr(C)*	*Pr(¬C)*
1	0.70	0.30
0	0.00	1.00

Pr(D)	*Pr(¬D)*
0.70	0.30

Fig. 3. Simple BAG illustrating probability computations.

In Fig. 3, the unconditional probability at node A is derived as the joint probability of A along with all nodes that influence its outcome, which is essentially all ancestors of A. We compute $Pr(A) = 0.6060 \approx 61\%$.

Similarly, unconditional probabilities at nodes B and C can be computed by considering the sub-network rooted at the corresponding nodes. The unconditional probabilities are shown under the LCPD table of each node. Fig. 2 shows the unconditional probabilities of the nodes in our test network. The security administrator can use this threat model to prioritize risk and derive an effective security hardening plan so as to reduce the risk to a certain level (e.g. $< 50\%$) before deploying the system.

4.4 Posterior probability with attack evidence

The BAG is next used to address dynamic aspects of the security planning process. Every network state has a certain probability of occurrence. This probability can change during the life time of the system due to emerging security conditions, changes in contributing factors or the occurrence of attack incidents. The BAG can then be used to calculate the posterior probabilities in order to evaluate the risk from such emerging conditions.

Let $S = \{S_1, \ldots, S_n\}$ be the set of attributes in a BAG and $E = \{S'_1, \ldots, S'_m\} \subset S$ be a set of attributes where some evidence of exploit have been observed. We can say that attributes in E are in the *true* state, i.e. $S'_i = 1$ for all $S'_i \in E$. Let $S_j \in S - E$ be an attribute whose posterior probability has to be determined. The probability we are interested in is $Pr(S_j \mid E)$, and is obtained by using the Bayes Theorem:

$$Pr(S_j \mid E) = Pr(E \mid S_j) \times Pr(S_j)/Pr(E). \qquad (4.3)$$

$Pr(E \mid S_j)$ is the conditional probability of joint occurrence of $S'_1,...,S'_m$ given the states of S_j. $Pr(E)$ and $Pr(S_j)$ are the prior unconditional probability values of the corresponding attributes. Since evidence attributes in E are mutually independent, $Pr(E \mid S_j) = \prod_i Pr(S'_i \mid S_j)$ and $Pr(E) = \prod_i Pr(S'_i)$. For example, in Fig. 3, assume that the system administrator detects an attack incident on A (attacker compromises FTP server). The posterior probability of C is then computed as follows.

$$\begin{aligned}
Pr(C \mid A) &= Pr(A \mid C)Pr(C)/Pr(A) \\
&= 0.81 \quad \text{where} \\
Pr(A \mid C) &= \textstyle\sum_{B\in\{T,F\}} [Pr(A \mid B, C = T)Pr(B)] \\
&= (1.0 \times 0.6)_T + (1.0 \times 0.4)_F \\
&= 1.0 \\
Pr(A) &= 0.61 \\
Pr(C) &= 0.49
\end{aligned}$$

Similarly, the posterior probability at node B can be computed. Note that the unconditional probability of node C was originally 0.49. After taking into account the attack incident at node A, the posterior probability becomes 0.81. Further, computation of posterior probabilities for successor nodes of A (forward propagation) remains the same as described in the previous sub-section, with the change that the LCPDs at those nodes only account for the $A = 1$ case. In this manner, the security administrator can *hypothesize* the probability of occurrence of other nodes in the graph to respond to an emerging attack incident. Fig. 2 shows the posterior probabilities in response to two hypothetical evidences (denoted by the label Ⓔ) in the Mail server of our test network. Note that the parent ("*root access @ 196.216.0.19*") of the evidence node "*squid port scan*" has a posterior probability of less than 1.0. Ideally, given the evidence that the port scan has been executed, the attacker must have had root access on the machine. Hence, the parent node should also have an updated probability of 1.0. However, this inference assumes that the squid port scan is executable only after gaining root access on the machine. The system administrator may decide to relax such an assumption in order to account for uncertainties (e.g. zero-day attacks), achieved by replacing the zero values in Def. 5 with non-zero values. Such a relaxation will reduce the impact of the evidence nodes on their parents.

As seen in Fig. 2, most of the unconditional probabilities increase after the attack incidents, but not at the same rate. It is possible to have nodes with decreased probabilities as well. In this specific scenario, there is a significant increase in the chance that the administrator machine is targeted by an attacker. This observation shows that the attacker is likely to execute an attack to compromise the root machine. Hence, sufficient measures should be taken to protect it. Moreover, it is also possible that the mitigation plan designed earlier in static analysis may no longer be appropriate under the light of the emerging events. We will formally address this problem in the next section.

5 Security Risk Mitigation with BAG

Although many researchers have studied risk assessment schemes, including the NIST, the methodologies used to estimate loss varies from organization to organization. Loss can be measured in terms of monetary units, relative magnitudes [15], [16], [17], [18] or multi-units [13], [19], [20]. In a BAG, the security manager can choose to evaluate the risks by considering an expected loss/gain quantity. The expected loss/gain is computed from organization specific factors such as potential loss or gain associated with an attribute's states. It usually reflects the impact of attack likelihoods on the economic turnout of an organization. We will describe this scheme later in the section. We begin with the notion of a security control in the context of the BAG.

5.1 Assessing security controls

Definition 6: Security Control: Given a BAG $(S,\tau,\varepsilon,\mathcal{P})$, a Bernoulli random variable M_i is a security control if $\exists S_j \in N_{internal} \cup N_{external}$ such that $Pr(S_j \mid Pa[S_j], M_i = T) < Pr(S_j \mid Pa[S_j], M_i = F)$ for at least one assignment of states to $Pa[S_j]$. Further, $Pr(M_i) = 1.0$ if $M_i = T$; otherwise zero.

In other words, a security control is a preventive measure that minimizes or eliminates the likelihood of attack on one or more attributes so as to prevent an attacker from reaching its goal. We define the security measure as a Bernoulli random variable with the *true* state signifying that the control is enforced and *false* signifying that the control is known but not enforced. Enforcement of a control directly influences the LCPD of the associated attribute and indirectly impacts the unconditional probabilities of other attributes. For example, the probability of the node A in Fig. 3 is initially $Pr(A \mid B,C)$. Assume that a security measure M_0:*"local access control"* can influence the outcome at A. The probability distribution therefore becomes $Pr(A \mid B,C,M_0)$ and the LCPD of the node is expanded as shown in Table 2. The probabilities when $M_0 = 0$ are directly taken from the original LCPD. However, probabilities for $M_0 = 1$ are assigned based on certain subjective belief on the security measure's capacity to prevent the attribute's compromise. The modified LCPDs are used to compute the unconditional probability of nodes in the graph. It is not difficult to see that the unconditional probability of a node (and its successors) influenced by a control will reduce when the control is enforced. Note that, by definition, the unconditional probability of the control itself is 1.0 if its state is *true*.

Definition 7: Security Mitigation Plan: Given set $M = \{M_1,\ldots,M_m\}$ of m security controls, a security mitigation plan is represented by a boolean vector $\vec{T} = (T_1, T_2, \ldots, T_m)$ where $M_i = T_i$.

Therefore, the mitigation plan is a specification of which controls have been chosen for enforcement as part of the hardening process. Further, for a given control M_i,

TABLE 2
EXPANDED LCPD OF NODE A (IN FIG. 3) UNDER THE PRESENCE OF SECURITY CONTROL M_0.

B	C	$Pr(A)$	$Pr(\neg A)$
1	1	1.00	0.00
1	0	0.65	0.35
0	1	1.00	0.00
0	0	0.00	0.00

$\longrightarrow$

B	C	M_0	$Pr(A)$	$Pr(\neg A)$
1	1	1	0.50	0.50
1	1	0	1.00	0.00
1	0	1	0.65	0.35
1	0	0	0.65	0.35
0	1	1	0.75	0.25
0	1	0	1.00	0.00
0	0	1	0.00	1.00
0	0	0	0.00	1.00

consider the set $\mathcal{I}$ of all $S_j \in N_{internal} \cup N_{terminal}$ such that $Pr(S_j \mid Pa[S_j], M_i = T) < Pr(S_j \mid Pa[S_j], M_i = F)$ (for some assignment of states to $Pa[S_j]$). Then, the subset $\{S_k \in \mathcal{I} \,|\, Pa[S_k] \cap \mathcal{I} = \phi\}$ is called the *coverage* of M_i. With reference to Fig. 3, a control such as M_0*:"disconnect from Internet"* directly changes the probability $Pr(D)$ (equal to zero if $M_0 = 1$). This in effect changes the LCPD tables at nodes B, C and D. Therefore, the set $\mathcal{I}$ contains all four nodes for M_0. However, only node D is in the coverage of M_0 since, for all other nodes, one or more parent nodes are also present in $\mathcal{I}$. Intuitively, the coverage nodes are those whose LCPDs are directly affected by a security control, rather than by indirect inference. For a given security mitigation plan $\vec{T}$, we can define the plan coverage by collecting the coverage of each enforced control in the plan. Each control M_i also has an associated cost C_i of implementation, giving us the total plan cost as

$$SCC(\vec{T}) = \sum_{i=1}^{m} (T_i C_i). \tag{5.1}$$

5.2 Assessing security outcomes

When using a BAG, a better quantitative representation of the loss/gain is obtained by considering the expected loss/gain once a set of security measures have been implemented. Hence, we augment the BAG with a value signifying the amount of potential loss/gain at each node, and account for the security decision during evaluation.

Definition 8: AUGMENTED-BAYESIAN ATTACK GRAPH: Let $BAG = (S, \tau, \varepsilon, \mathcal{P})$ be a Bayesian attack graph. An augmented-Bayesian attack graph (augmented-BAG) $BAG_{aug} = BAG|(M, \gamma, V)$ is obtained by adding a node set M, edge set γ and by associating a value V_j to each $S_j \in S$, where

1) M is the set of security controls,
2) $\gamma \subseteq M \times S$. An ordered pair $(M_i, S_j) \in \gamma$ if S_j is in the coverage of M_i, and
3) V_j is the expected loss/gain associated with the attribute $S_j \in S$.

The set M extends the BAG with additional nodes representing hardening measures. The set γ represents the new edges between the controls and attributes of the graph. A new edge is inserted if a control directly influences the state of an attribute. In this work, all external attributes are given a zero cost, i.e. $V_j = 0$ for all $S_j \in N_{external}$. The value associated with $S_j \in N_{internal} \cup N_{terminal}$ is computed as

$$V_j = [1 - Pr(S_j)] \times G_j - Pr(S_j) \times L_j, \tag{5.2}$$

where L_j is the potential loss representing the damage value that an organization might have to pay when the attribute S_j is compromised, G_j is the potential gain if S_j is not compromised and $Pr(S_j)$ is the unconditional probability of S_j. If there exists $(M_i, S_j) \in \gamma$, $Pr(S_j)$ is computed as a conditional probability $Pr(S_j \mid M_i)$ where the state of M_i depends on the security plan $\vec{T} = (T_i)$. The expected loss/gain w.r.t. the security plan $\vec{T}$, denoted by $LG(\vec{T})$, is computed as the cumulative sum of all node values, i.e.

$$LG(\vec{T}) = \sum_{S_j \in S} V_j. \tag{5.3}$$

A positive value of LG signifies gain, while a negative value signifies loss. Note that we do not assume any particular cost model in our formulations, both for control cost and loss/gain evaluation. The cost model is usually subjective to organizational policies and hence can differ from one institution to another. The cost factors considered here (security control cost, potential loss and potential gain) are standard quantities that any organization must be able to determine in order to perform risk analysis.

5.3 Assessing the security mitigation plan

In order to defend against the attacks possible, a security manager can choose to implement a variety of safeguard technologies, each of which comes with different cost and coverage. For example, to defend against the "*ftp/.rhost*" exploit, one might choose to apply a security patch, firewall, or simply disable the FTP service. Each choice of action has a different cost and different outcome. A security administrator has to assess the technologies and make a decision towards maximum resource utilization. The two objectives we consider in this study are the total security control cost and the expected loss/gain. The single-objective problem is the most likely approach to be taken by a decision maker. The problem is stated as follows.

The Single-Objective Optimization Problem (SOOP): *Given an augmented-BAG* $(S, \tau, \varepsilon, \mathcal{P}) \mid (M, \gamma, V)$*, find a vector* $\vec{T}^* = (T_i^*)$, $T_i^* \in \{0, 1\}; 1 \le i \le |M|$*, which maximizes the function* $\alpha LG(\vec{T}^*) - \beta SCC(\vec{T}^*)$*, where* α *and* β *are preference weights for the expected loss/gain and the total cost of security control respectively,* $0 \le \alpha, \beta \le 1$ *and* $\alpha + \beta = 1$.

The method for assessing a security plan is as follows. First, the security analyst chooses a subset of security controls to construct a security plan $\vec{T}^*$. She then updates the unconditional probability of all attributes using the plan coverage information. She computes the expected loss/gain associated with every attribute $S_j \in S$ using Eq. (5.2). Finally, the total expected loss/gain of the entire graph is taken as an assessment of the security plan's outcome. The best security plan is the one that maximizes the function $\alpha LG(\vec{T}^*) - \beta SCC(\vec{T}^*)$.

Given only two objectives, a preference based single-objective approach might seem to provide a solution in accordance with general intuition. However, the quality of the solution obtained using this process can be quite sensitive to the assignment of the weights. In addition, security administrators often have to work within a limited budget that may be less than the minimum cost of system hardening. The objective in such a case is to design a security plan that maximizes the organization's financial throughput.

The next level of sophistication is added by formulating the optimization as a multi-objective problem. The multi-objective approach alleviates the requirement to specify any weight parameters and hence a better global picture of the solutions can be obtained.

The Multi-Objective Optimization Problem (MOOP): *Given an augmented-BAG* $(S,\tau,\varepsilon,\mathcal{P}) \mid (M,\gamma,V)$*, find a vector* $\vec{T}^* = (T_i^*)$*,* $T_i^* \varepsilon \{0,1\}; 1 \leq i \leq |M|$*, which minimizes SCC and maximizes LG.*

Solutions to a multi-objective problem are characterized by the concept of Pareto-optimality. In our case, a security plan $\vec{T}_1$ is *Pareto-optimal* if there is no other plan $\vec{T}_2$ such that

1) $SCC(\vec{T}_2) < SCC(\vec{T}_1)$ and $LG(\vec{T}_2) \geq LG(\vec{T}_1)$, or
2) $SCC(\vec{T}_2) \leq SCC(\vec{T}_1)$ and $LG(\vec{T}_2) > LG(\vec{T}_1)$.

If any of these conditions hold, then $\vec{T}_2$ is said to *dominate* $\vec{T}_1$. $\vec{T}_1$ and $\vec{T}_2$ are *non-dominated* w.r.t. each other if none dominates the other. Pareto-optimal solutions are non-dominated w.r.t. all other solutions. A multi-objective optimizer identifies (or approximates) the set of Pareto-optimal solutions and reveals the trade-off relations between the underlying objectives. Choice of a final solution from this set is at the discretion of the decision maker, often decided by the cost to benefit ratio.

For the BAG shown in Fig. 4, we have identified that 13 different security controls are possible. These controls are represented by an 'ellipse' in the figure. These security controls produce 2^{13} candidate security plans. A genetic algorithm based approach is presented next to search through these candidate plans in an efficient manner.

5.4 Genetic algorithm

The genetic algorithm used in the study begins with a population P_0 of N randomly generated security plans. A generation index $t = 0,1,\ldots,Gen_{MAX}$ keeps track of the number of iterations of the algorithm. Each iteration proceeds as follows. The *SCC* and *LG* values of every plan in P_t are calculated. $N/2$ plans are then selected from P_t to form a mating pool M_t. The selection process is different for SOOP and MOOP, and discussed later. An offspring population Q_t (containing $N/2$ plans) is generated from the mating pool by using the standard single-point binary crossover and mutation operators [21]. The process is then repeated with $P_{t+1} = Q_t \cup M_t$ until $t = Gen_{MAX}$.

5.4.1 Solving SOOP

The selection process to solve SOOP is based on the objective function $\alpha LG(\vec{T}) - \beta SCC(\vec{T})$ and uses the process of binary tournament. In this process, two plans are randomly selected (with replacement) from P_t and the one with the higher objective function value is inserted into the mating pool. This process is repeated until the mating pool is full. The solution to SOOP is the plan with the highest objective value across all iterations of the algorithm.

5.4.2 Solving MOOP

Simple objective value comparison is not possible in the presence of more than one objective function. Hence, a different selection scheme is required for MOOP. The scheme used here is based on *non-dominance ranking*, a popular concept in evolutionary multi-objective optimization. Under this process, all non-dominated solutions in P_t (solutions not dominated by any other solution in P_t) are identified and assigned a rank 1. The rank 1 solutions are then removed from P_t and the non-dominated solutions in the remaining population form rank 2 solutions. The process is repeated until all solutions are assigned a rank. Selection of $N/2$ solutions for the mating pool is then performed according to increasing rank. A *crowding distance metric* [22] is used if the number of solutions required to fill the mating pool is lower than the available solutions in the rank being considered. The crowding distance of a solution is the perimeter of the rectangle formed by its two neighbors of the same rank; the distance is infinite for points having less than two neighbors (e.g. extreme points). Choice of solutions within a rank is done in decreasing order of crowding distance, thereby giving preference to solutions that are not at close proximity to others. The set of solutions to MOOP are the rank 1 solutions of $P_{Gen_{MAX}}$.

6 Empirical Results

In the preparation phase, we conduct risk assessment analysis to initially compute the static risk. Fig. 2 shows the unconditional probabilities at the nodes of the test network. We identify 13 security controls that can be used to reduce the risk. We assign a security control cost to each individual control and link each control to the attribute(s) in the BAG that are covered by it. The augmented-BAG resulting from this process is shown in Fig. 4. Next, we assign different damage costs and revenue to every attribute in the graph. Although we do not assume any particular cost model and values are assigned hypothetically for the purpose of demonstration, we did try to maintain some relative difference in magnitude to account for the relative importance of different services.

In the first experiment, we assess the expected loss/gain on top of the static risk analysis results (Fig. 2) using Eq. (5.2). When using no security control, i.e. a mitigation plan signified by the zero vector, we have an overall expected gain of 622.0 units. We then assess the cost on the dynamic environment where we assume that two attack incidents have been detected. Fig. 2 and Fig. 4 show the posterior probabilities and the expected loss/gain at the nodes under this situation. Note that these attack incidents quickly change the business scenario. The total expected loss/gain (LG) changes from 622.0 to -398.17 units. We also notice a change in the momentum of risk. In particular, the posterior probabilities indicate a significant change in

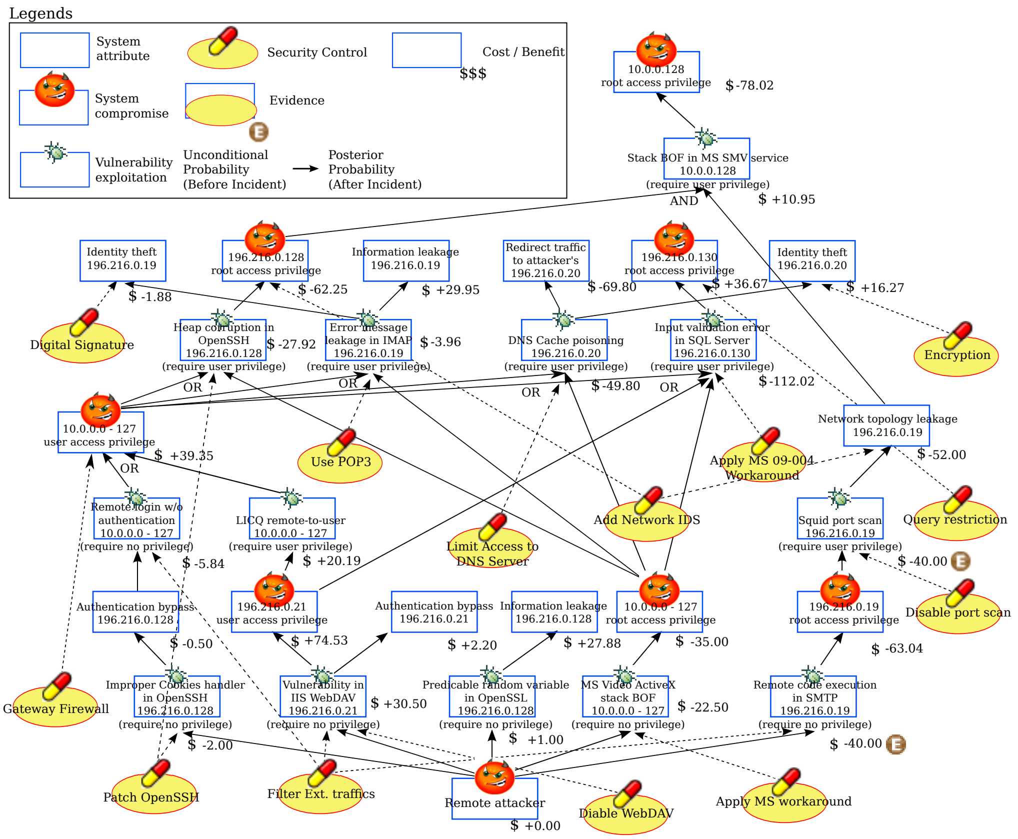


Fig. 4. Augmented-BAG of test network under two attack incidents with 13 security controls. The expected loss/gain (V_j) is shown at each node.

TABLE 3
SECURITY OUTCOME ASSESSMENT FOR EACH INDIVIDUAL CONTROL IN AUGMENTED-BAG OF TEST NETWORK.

Security Control	Cost ($\mathcal{A}$)	Outcome ($\mathcal{B}$)	Net benefit ($\mathcal{B} - \mathcal{A} - 622.0$)
apply OpenSSH security patch	63	1407.89	722.89
apply MS work around	14	1202.45	566.45
filtering external traffic	70	1208.84	516.84
limit DNS access	53	1000.65	325.65
disable portscan	11	875.44	242.44
disable WebDAV	250	1095.90	223.90
apply MS09-004 work around	31	861.10	208.10
add Network IDS	102	858.91	134.91
add Firewall	205	881.15	54.15
encryption	34	681.75	25.75
digital signature	33	673.281	8.28
query restriction	84	681.00	- 25.00
use POP3 instead	153	704.67	- 70.33

risk level at the Administrative server owing to the two attack incidents. This change influences the priority of risks identified earlier during static analysis, and highlights the importance of dynamic risk analysis.

Next, we conduct several tests to assess the outcome of using a security control. The base case where no individual control is used yields an expected gain of 622.0 units. Table 3 shows the net benefit of using each control individually. At this point, the security administrator may want to rank the security outcomes and build a security mitigation plan from the top-ranked controls. Such a methodology has two drawbacks.

First, the ranking procedure itself is not straight forward because of reciprocal relationships between control cost and expected outcome. For example, “*disable portscan*” and “*filtering external traffic*” when applied alone raises the expected gain from 622.0 units to 875.44 (an increase of 253 units) and 1208.84 units (an increase of 587 units), respectively. The combined outcome when applying both is 1351.27 units (less than $622+253+587$). On the other hand, combining “*add Firewall*” (individual increase from 622.0 to 881.15 units) and “*apply MS work around*” (individual increase from 622.0 to 1202.45 units) can raise the outcome to 1735.6 units (greater than $622+259+580$). The latter two are better choices based on expected outcome, but the former two incurs a lower cost of implementation. This makes the ranking of controls, based on a specific cost factor, a difficult process. Second, even if a ranking has been established, greedy selection can lead to sub-optimal plans. Assume that controls are ranked based on the net benefit they incur individually. The security controls are ordered in this manner in Table 3. Given a control cost constraint of, say, 200.0 units and a selection scheme based on the ranks, an administrator will choose the first four controls in the table. These controls have a combined cost of 200.0 units and result in an expected gain of 2673.96 units (a net benefit of 2473.96 units collectively). However, selecting the 5th and the 7th controls, instead of the 4th one, effectuates an expected gain of 2809.28 units at the cost of 189.0 units (a net benefit of 2620.28 units). This shows that the security administrator should not choose the security controls based on their individual outcomes or

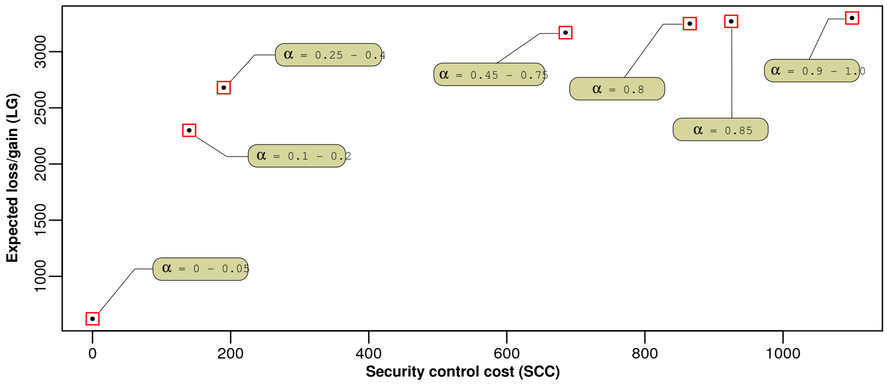


Fig. 5. Genetic algorithm solutions to single objective problem obtained by using different weights.

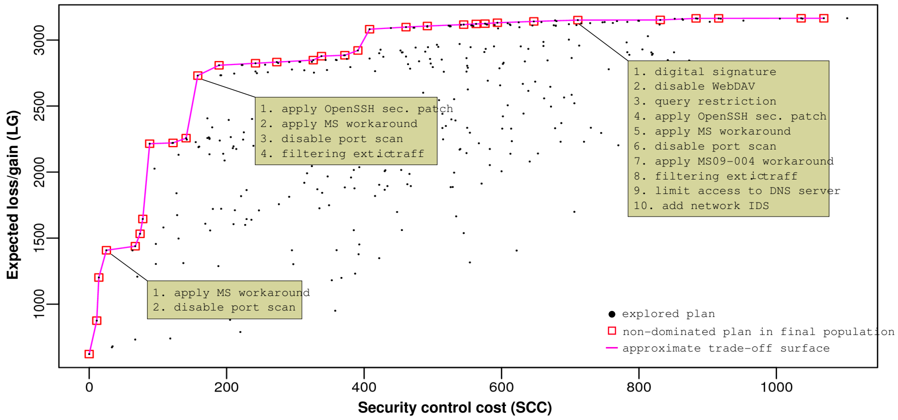


Fig. 6. Genetic algorithm solutions to multi-objective problem with static risk assessment.

by greedy selection. Instead, a more sophisticated decision making platform is required. This motivates the next three experiments with single and multi-objective optimization.

We conduct three risk mitigation analysis experiments on the test network. The genetic algorithm discussed in Section 5.4 is used for this analysis. The algorithm parameters are set as follows: population size $N = 100$, $Gen_{MAX} = 50$, crossover probability $= 0.8$, and mutation probability $=$ 0.01. We ran each instance of the algorithm five times to check for any sensitivity of the solutions obtained from different initial populations. We also check if running the algorithm for a higher number of iterations (upto 200 generations) results in any improved convergence. However, since the solutions always converged to the same optima (or set of optima), we dismiss the presence of such sensitivity.

In single-objective cost analysis, we run multiple instances of SOOP using different combination of values for α and β. α is varied in the range of $[0,1]$ in steps of 0.05. β is always set to $1-\alpha$. Fig. 5 shows the solutions obtained from this process. In general, a decision maker may want to assign equal weights ($\alpha = 0.5$) to both objective functions – security control cost and total expected loss/gain. It is clear from the figure that such an assignment does not necessarily provide the desired balance between the two objectives. Furthermore, the solutions are quite sensitive to the weights and they are not uniformly distributed across different ranges of α. Since the weights do not always influence the objectives in the desired manner, understanding their effect is not a trivial task for the administrator. It is also not always possible to perform an exhaustive analysis of the affect of the weights on the objectives. Given such situations, the decision maker should consider obtaining a global picture of the trade-offs possible. With such a problem in mind, we next consider the multi-objective variant.

Fig. 6 shows the non-dominated solutions (in $P_{Gen_{MAX}}$) obtained in the multi-objective analysis. Further, all mitigation plans explored by the genetic algorithm during the iterations are highlighted. The algorithm reported all solutions generated for SOOP (using multiple α), as well as several others, specifically solutions in the range where the security control cost is between 200.0 and 700.0 units. These new solutions provide much better flexibility in the decision making process. Moreover, performing the multi-objective analysis is much faster than solving SOOP. This is because the security administrator has to solve SOOP with multiple parameter settings in order to identify the plan with the desired outcomes, whereas by solving MOOP, one can generate a good overview of multiple plans in one single run.

In the last experiment, we use the genetic algorithm to assess the choice of security hardening in a dynamic environment. Fig. 7 shows the choices of mitigation plans in response to two emerging attack incidents, previously shown in Fig. 4. In this plot, we compare the dynamic

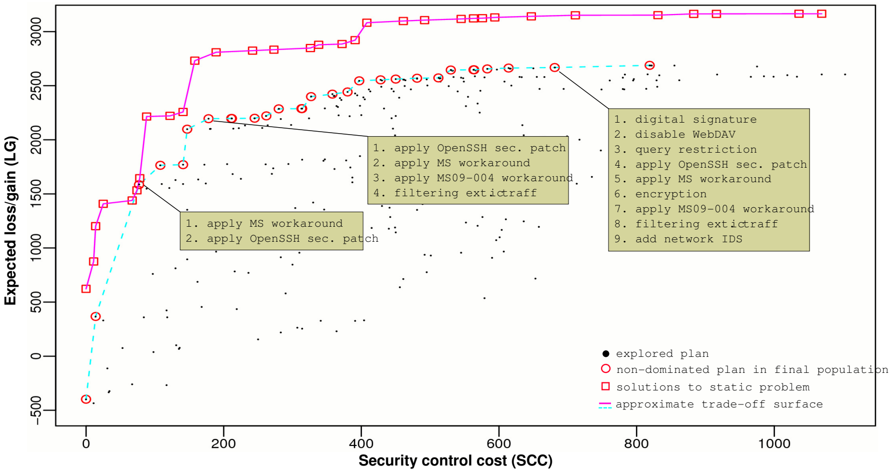


Fig. 7. Genetic algorithm solutions to MOOP with dynamic risk assessment.

results with the static ones. Not surprisingly, the plans in this case effectuate lower gains owing to the damage already caused by the attacker when (and at which point) the incidents are detected. Despite this difference, the mitigation plans with similar costs are not so different between the static and dynamic solutions. The three plans highlighted in the figure are very similar to those shown in Fig. 6. Such minimal changes in plan characteristics can be considered a positive outcome since the security administrator is not required to revise the entire plan chosen during static analysis. Instead, she can exploit the commonalities for efficient usage of already invested resources. Results from the dynamic analysis also highlight the requirement for pro-active action in security management. Note that although not implementing any controls still results in a positive gain, the appearance of two attack incidents quickly transform this into a case with negative expected outcome.

7 Related works

Attack graphs have been studied in several areas of security risk management. [26], [27] propose an attack graph-based probabilistic metric model to quantify the overall security of network system. In this paper, attack graph is used to represent the casual relationship between vulnerabilities encoded in the attack graph. Similar to the Bayesian attack graph model, a node in attack graph is assigned with an intrinsic score representing the likelihood of vulnerability exploitation but the final probability of success in that node is computed by conjunctive probability or disjunctive probability. The authors focus their efforts in solving the problem of cycles in attack graph. Although cycles can occur in our Bayesian attack graph model, we argue that such cycles can be disregarded. As a result we are able to focus on other applications of attack graph analysis in addition to those proposed by [27].

Wang et al. [30] extend attack graph analysis to intrusion detection. In this model, attack graphs are pre-generated, and then used as a knowledge base for correlating received alerts, hypothesizing missing alerts and predicting future alerts. In [29], attack graphs are used to pre-identify attack paths. Knowing the paths in which security can be breached, the authors use a greedy algorithm to identify optimal sensor placements that provide a complete monitoring coverage of all potential attack paths. Although it appears that the authors overlook the robustness issue, this approach is a good application of an attack graph in risk management analysis. Frigault and Wang [31] use Bayesian networks with attack graphs to statically analyze the inherent risk in a network. Frigault et al. [32] introduce a Dynamic Bayesian Networks based model to incorporate temporal factors, such as, how vulnerabilities evolve over time in the attack graph. In this manner, they model the security of dynamically changing networks. Xie et al. [33] also use Bayesian networks for security risk analysis of networked systems. They incorporate run-time observations like IDS alerts to compute security risks. Our Bayesian attack graph model is similar to these works, although much richer. Our threat modeling combines asset identification, system vulnerability and connectivity analysis, and mitigation strategies. Dantu et al. [10], [34], [35] also use Bayesian networks for security risk management. Their model is fundamentally different from ours in the sense that they focus on computing the probability of attacks based on attacker capabilities and behavior. This is an important contribution that is worth investigating further.

Minimization analysis has been thoroughly studied by several research groups [12], [23], [28], [29]. In minimization analysis, the attack graph model is rewritten in terms of a Conjunctive Normal Form (CNF). The minimum security hardening is obtained from the negation of CNF w.r.t. the proposed cost model. Most minimization analysis approaches scope the problem domain to network configurations. Our multi-objective analysis, on the other hand, extends the problem domain to cover the use of external security controls. Hence, our risk mitigation analysis gives a more comprehensive solution than traditional minimization analysis. In fact, we compare the multi-objective analysis with [12], the pioneer work in this area. We have found that our optimal solutions include the minimal security hardening solution. Although a GA is not always guaranteed to find the minimal solution every time it runs, it can provide choices of implementation, allowing security analysts to comprehend the pros and cons of each option.

The practical use of attack graphs has been studied by Saha [25]. In this paper, attack graph analysis is used in real network practice. The paper also addresses the practical problem of maintaining an attack graph in response to the changes in network configurations. Toward this problem, the author proposes an incremental approach where changes in network configuration are modeled as node insertion and node deletion. The benefit of an incremental algorithm is that the analyzer can avoid attack graph regeneration from scratch. It will be interesting for us to study such a technique to improve our attack graph analysis tool.

8 Conclusion

In this paper, we address the system administrators' dilemma, namely, how to assess the risk in a network system and select security hardening measures from a given set of controls so as to maximize resource utilization. One important contribution of our solution methodology is the use of a BAG model of the network to drive the decision process. We have provided formal definitions for network characteristics, attacks and security measures under this model. We also show that by using a BAG, we are able to better understand the causal relationships between preconditions, vulnerability exploitations and post conditions. This is facilitated by computing the likelihoods of different outcomes possible as a result of the cause-consequence relationships. We have demonstrated how the BAG can be used to revise these likelihoods in the event of attack incidents. Using empirical results on a test network, we show that such a dynamic risk analysis helps the system administrator identify evolving weak spots in a network. We also provide the necessary optimization formulations required to build a mitigation plan that reduces the risk levels. Towards this end, we propose a genetic algorithm capable of performing both single and multi-objective optimization of the administrator's objectives. While single objective analysis uses administrator preferences to identify the optimal plan, multi-objective analysis provides a complete trade-off information before a final plan is chosen. Results are shown to demonstrate the effectiveness of the algorithm in both static and dynamic risk mitigation.

As immediate future work, we shall work on improving the scalability and efficiency of our methodology. There are primarily three components in our approach that impact its scalability.

1) the creation of the Bayesian attack graph for the enterprise being evaluated,
2) the computation of the marginal probabilities in the Bayesian attack graph, and
3) the solution for the optimization problems.

Attack graphs for large networks can get complex. We have the search space bound to the number of attribute-instances that specify what vulnerabilities are present in which machines. The size of the attribute-instances can be as large as A × M, where A is the number of attributes and M be the number of machines in the system. The monotonicity assumption (see Section 4.3) ensures that each attribute-instance is used only once. Hence the search space is reduced. So given N = A × M, the graph generation is bounded to O(N^2) on each branch. As the number of branches is bounded to N, the overall algorithm will never go beyond O(N^3). Note that the generation of the attack graph is a one-time cost and is not done in real time. Thus, the generation of the attack graph does not seriously impact the performance.

The computation of the marginal probabilities in the BAG is of bigger concern. The evaluation algorithm is used to compute the unconditional probabilities and is currently implemented using brute force DFS traversal. Posterior probability computation is expensive using this implementation and therefore impacts the decision making time in a dynamic scenario. Consider the chain rule of equation (4.2). Since each variable S_i can take true or false, BBN has an exponential complexity. In other words, for a given set S of n variables, there are 2^n cases for computing $Pr(S_1,...,S_n) = \prod_{i=1}^{n} Pr(S_i \mid Pr(Pa[S_i]))$, the marginal probabilities. Hence, computing marginal probabilities either for prior or posterior cases, is O(2^n) and does not scale very well for a large network. However, there are more efficient algorithms (such as the one using Monte Carlo approximation proposed by Santos and Shimony [36]) that can give a fair approximation for these probabilities. We plan to look into how these approximations can be used to speed up the analysis and what the corresponding impact is on the accuracy of the solution. In particular, we wish to revise the evaluation algorithm to include heuristic based update mechanisms in order to reduce the time required to complete the mitigation analysis, without scarifying the quality of results obtainable. Furthermore, the mitigation process in dynamic situations needs to be improved so that a security administrator can quickly identify the best security response that accounts for all former investments made as part of the static analysis stage.

Last, but not least, evolutionary algorithms often receive criticism for their time complexity, compared to other optimization methods. The MOOP version of the genetic algorithm used in this study has a complexity of O(GNlogN), where G is the number of generations and N is the population size. However, the population based approach also makes it highly suitable for discovering multiple solution points on the Pareto-front. Further, these algorithms are inherently parallel and can easily be adapted to utilize the processing power of most massively parallel systems [37]. Nonetheless, we are seeking to evaluate multiple solutions during the run of the algorithm, and, hence, complexity of the BAG calculation is a crucial component of the optimization. The evolutionary algorithm is one viable methodology for the multi-objective optimization that we can think of at this moment. There is no doubt that more efficient methods are required. We strongly believe this would motivate some future studies in this area.

It is worth mentioning that some security controls have been found to be commonly included in the optimal solutions. It is possible that security hardening is more critical in certain areas of the attack graph. Such areas could be nodes that has multiple fan outs. In other words, these critical areas are at-risk junctions that can be used by an attacker to cause multiple outcomes. Security controls that can reduce risk in such areas are likely to be parts of the optimal solutions. Therefore, it is worth investigating how such controls can be identified efficiently so as to reduce the search space for the optimization algorithm.

Acknowledgment

This work was partially supported by the the National Science Foundation under grant no. 0905232 and by the Air Force Office of Scientific Research under grant no. FA 9550-07-1-0042. We gratefully acknowledge their support.

References

[1] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the Ninth Conference on Computer and Communications Security*, pages 217–224, Washington, DC, 2002.

[2] S. Jha, O. Sheyner, and J. M. Wing. Two formal analysis of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pages 49–63, Cape Breton, Nova Scotia, Canada, 2002.

[3] C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the New Security Paradigms Workshop*, pages 71–79, Chicago, IL, 1998.

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

[5] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer attack graph generation tool. In *Proceedings of the 2nd DARPA Information Survivability Conference and Exposition*, pages 307–321, Anaheim, CA, 2001.

[6] J. Dawkins, C. Campbell, and J. Hale. Modeling network attacks: extending the attack tree paradigm. In *Proceedings of the Workshop on Statistical Machine Learning Techniques in Computer Intrusion Detection*, Johns Hopkins University, Baltimore, MD, 2002.

[7] A. P. Moore, R. J. Ellison, and R. C. Linger. Attack modeling for information survivability. Technical Note CMU/SEI-2001-TN-001, Carnegie Melon University / Software Engineering Institute, March 2001.

[8] I. Ray and N. Poolsappasit. Using attack trees to identify malicious attacks from authorized insiders. In *ESORICS 2005*, pages 231–246, Milan, Italy, 2005.

[9] B. Schneier. Attack Trees. *Dr. Dobb's Journal*, December 1999.

[10] R. Dantu, K. Loper, and P. Kolan. Risk management using behavior based attack graphs. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, pages 445–449, Las Vegas, NV, 2004.

[11] Y. Liu and H. Man. Network vulnerability assessment using Bayesian networks. In *Proceedings of the SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, volume 5812, pages 61–71, Orlando, FL, 2005.

[12] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 86–95, Las Vegas, NV, 2003.

[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 14th ACM conference on Computer and communications security*, pages 204–213, Alexandria, VA, 2007.

[14] M. Schiffman. Common Vulnerability Scoring System (CVSS). http://www. first. org/cvss/cvss-guide. html.

[15] W. Lee. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10(1):5–22, 2002.

[16] G. Stoneburner, A. Goguen, and A. Feringa. Risk management guide for information technology systems. *NIST Special Publication*, pages 800–30, 2002.

[17] B. Berger. Data-centric quantitative computer security risk assessment. SANS Institute InfoSec Reading Room, 2003.

[18] A. Arora, D. Hall, C. A. Piato, D. Ramsey, and R. Telang. Measuring the risk-based value of IT security solutions. *IT Professional*, 6(6):35–42, 2004.

[19] S. A. Butler. Security attribute evaluation method: A cost-benefit approach. In *Proceedings of the 24rd International Conference on Software Engineering*, pages 232–240, Orlando, FL, 2002.

[20] S. A. Butler and P. Fischbeck. Multi-attribute risk assessment. In *Proceedings of SREIS02 in conjunction of 10th IEEE International Requirements Engineering Conference*, Raleigh, NC, 2002.

[21] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[23] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, November 2006.

[24] X. Ou, S. Govindavajhala, and A. W. Appel. Mulval: a logic-based network security analyzer. In *Proceedings of the 14th conference on USENIX Security Symposium*, pages 113–128, Baltimore, MD, 2005.

[25] D. Saha. Extending logical attack graph for efficient vulnerability analysis. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 63–73, Alexandria, VA, 2008.

[26] L. Wang, A. Singhal, and S. Jajodia. Measuring the overall security of network configurations using attack graphs. In *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Application Security*, pages 98–112, Redondo Beach, CA, 2007.

[27] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In *Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 283–296, London, U.K., 2008.

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

[29] S. Noel and S. Jajodia. Optimal IDS sensor placement and alert prioritizing using attack graphs. *Journal of Network and Systems Management*, 16(3):259–275, September 2008.

[30] L. Wang, A. Liu, and S. Jajodia. Using attack graph for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29(15):2917–2933, November 2006.

[31] M. Frigault and L. Wang. Measuring network security using Bayesian network-based attack graphs. In *Proceedings of the 32nd Annual IEEE International Computer Software Applications Conference*, pages 698–703, Turku, Finland, 2008.

[32] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic Bayesian network. In *Proceedings of the 14th ACM Workshop on Quality of Protection*, Alexandria, VA, 2008.

[33] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy. Using Bayesian networks for cyber security analysis. In *Proceedings of the 40th IEEE/IFIP International Conference on Dependable Systems and Networks*, Chicago, IL, 2010.

[34] R. Dantu, P. Kolan, R. Akl, and K. Loper. Classification of attributes and behavior in risk management using Bayesian networks. In *Proceedings of the IEEE Intelligence and Security Informatics Conference*, pages 71–74, New Brunswick, NJ, 2007.

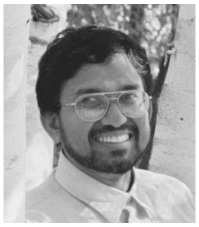
[35] R. Dantu, P. Kolan, and J. Cangussu. Network risk management using attacker profiling. *Security and Communication Networks* 2:83–96, 2009.

[36] E. J. Santos and S. E. Shimony. Exploiting case-based independence for approximating marginal probabilities. *International Journal of Approximate Reasoning*, 14(1):25–54, January 1996.

[37] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, October 2002.

Nayot Poolsappasit received his Ph.D. from Colorado State University in 2010. He is currently a post doctoral research fellow at the Missouri University of Science and Technology. He performs scientific research in security risk assessment and trusted computing in sensor networks. His current research interests include sensor-cloud services, trusted data aggregation, and identity and access management in virtual sensor networks. He is a member of the IEEE.

Rinku Dewri is an Assistant Professor in the Computer Science Department at University of Denver. He obtained his Ph.D. in Computer Science from Colorado State University. His research interests are in the area of information security and privacy, risk management, data management and multi-criteria decision making. He is a member of the IEEE and the ACM.



Indrajit Ray is an Associate Professor in the Computer Science Department at Colorado State University. Prior to that, he worked as an Assistant Professor in the Computer and Information Science Department at the University of Michigan-Dearborn. His main research interests are in the areas of computer and network security, database security, security and trust models, privacy and computer forensics. He is on the editorial board of several journals, and has served or is serving on the program committees of a number of international conferences. He is a member of IEEE, IEEE CS, ACM, ACM SIGSAC, IFIP WG 11.3 and IFIP WG 11.9.