Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Exploring attack graph for cost-benefit security hardening: A probabilistic approach

Shuzhen Wang [a,2], *Zonghua Zhang* [b,*,1], *Youki Kadobayashi* [c]

[a] *School of Computer Science, Xidian University, China*
[b] *Institute Mines-Telecom/TELECOM Lille 1, France*
[c] *NAIST, Japan*

ARTICLE INFO

Article history:
Received 15 July 2011
Received in revised form
15 August 2012
Accepted 23 September 2012

Keywords:
Security management
Vulnerability analysis
Risk assessment
Attack graph
Security hardening
Security metrics

ABSTRACT

The increasing complexity of today's computer systems, together with the rapid emergence of novel vulnerabilities, make security hardening a formidable challenge for security administrators. Although a large variety of tools and techniques are available for vulnerability analysis, the majority work at system or network level without explicit association with human and organizational factors. This article presents a middleware approach to bridge the gap between system-level vulnerabilities and organization-level security metrics, ultimately contributing to cost-benefit security hardening. In particular, our approach systematically integrates attack graph, a commonly used effective approach to representing and analyzing network vulnerabilities, and Hidden Markov Model (HMM) together, for exploring the probabilistic relation between system *observations* and *states*. More specifically, we modify and apply dependency attack graph to represent network assets and vulnerabilities (*observations*), which are then fed to HMM for estimating *attack states*, whereas their transitions are driven by a set of predefined cost factors associated with potential attacks and countermeasures. A heuristic searching algorithm is employed to automatically infer the optimal security hardening through cost-benefit analysis. We use a synthetic network scenario to illustrate our approach and evaluate its performance through a set of simulations.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

Theoretically, an exhaustive vulnerability searching and patching may lead to a secure system. This is a mission impossible in practice, however, due to the fact that the complexity and diversity of today's computer systems keep continuously increasing, introducing countless zero-day vulnerabilities. Also, a comprehensive vulnerability analysis usually costs much time, labor and resource, causing undesirable consequence to network and organization assets. A best practice is to prioritize the vulnerabilities and handle the key ones via cost-benefit analysis. This is extremely important in a resource-constrained network which has security budget limit. To that end, three issues must be carefully considered: (1) evaluating the criticality of individual vulnerabilities; (2) examining the implicit associations among a set of vulnerabilities exploited by an attack; (3) quantify the expected effectiveness of countermeasures.

To date, three issues have attracted much attention from both industry and academia. For instance, Common

* *Corresponding author*. Tel.: +33 320436422.
E-mail address: zonghua.zhang@telecom-lille1.eu (Z. Zhang).
[1] Most of the work reported in this paper was done when the author was with NICT, Japan. This paper subsumes a short version reported in (Zhang, 2012).
[2] The first author was financially supported by the National Natural Science Foundation of China under grant No. 61100156.

0167-4048/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2012.09.013

Vulnerability Scoring System (CVSS) has become a widely accepted industry standard, which rates severity and risk of individual vulnerabilities based on a suit of predefined security metrics and formulas (Mell et al., 2006). Also, a variety of graphical models, such as attack tree (Ray and Poolsappasit, 2005; Schneier, 1999) and attack graph (Ammann et al., 2002; Sheyner et al., 2002; Swiler et al., 2001), can serve as formal approaches to analyze the interdependency and causal relations between vulnerabilities, although their accuracy, adaptability, and scalability remain challenging. Recent years have witnessed increasing research attention to the interactions between system, human and organizations, ranging from security metrics (Jaquith, 2007) and security economics (Anderson and Moore, 2006) to security policies and cost-sensitive response (Kheir et al., 2010). One of the primary goals of these research is to facilitate security administrator's (SA for short) understanding on network threats and risk assessment, eventually achieving the most cost-effective security investment (Dewri et al., 2007; Noel et al., 2003). However, a systematic approach to bridging the gap between low-level vulnerability analysis and high-level security management has never been seen so far.

To bridge the aforesaid gap, we develop a middleware approach to explore AG-represented vulnerability information by using Hidden Markov Model (HMM). A holistic design framework is shown in Fig. 1, where our design is emphasized with bold blocks. In particular, we slightly modify dependency attack graph to represent network assets and vulnerabilities. HMM is then applied to capture the uncertainties of those *explicit observations* and estimate *attack states*, whose transitions are driven by a set of cost factors associated with potential attacks and security hardening. As a result, a probabilistic mapping between *network observations* and *attack states* can be established, and those *root-cause vulnerabilities* are indicated with higher probabilities. More generally, the interleaved two-tier model enables SAs to predict system evolution in the presence of vulnerabilities (both known and zero-day ones), identify the root causes (namely those significant observations) of attacks, and eventually take appropriate countermeasures. The probabilistic nature of AG-HMM determines that only heuristic algorithms can be used to infer the critical network

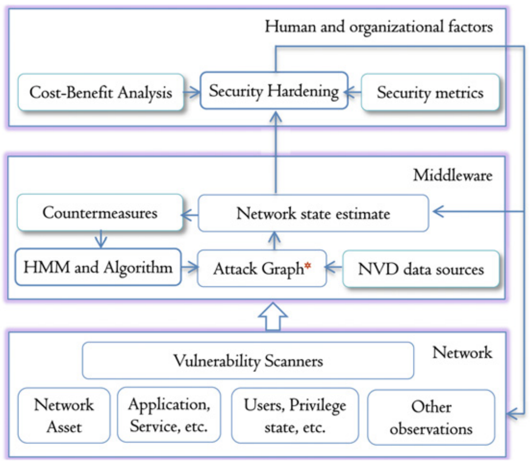

Fig. 1 – A holistic design perspective. Shown are our design focus (bold blocks and arrows).

states and cost-effective security hardening. To the best of our knowledge, this is one of the first attempts to develop a systematic approach to automatically take cost-effective security hardening based on vulnerability analysis.

The rest of this article is organized as follows. We review some relate work in Section 2. A motivating example is given in Section 3 for illustrating the design objective of our approach. We present our approach in details in Section 4, and report our experiments in Section 5. The article is then concluded in Section 6.

2. Related work

Our work is a cornerstone of three research topics: generation of attack graph, post-processing of attack graph, and its applications to risk assessment.

2.1. *Generation of attack graphs*

Attack graph, which is viewed as a useful method for network vulnerability analysis, has attracted much research effort in the last two decades (Lippmann and Ingols, 2005). Their construction generally falls into two categories: the first form is to represent network states as a whole in terms of known vulnerabilities and enumerate state transitions via model checking (Sheyner et al., 2002; Swiler et al., 2001); the second type is to combine and encode individual vulnerabilities by identifying their causal dependency (Ammann et al., 2002; Ingols et al., 2009; Ou et al., 2005). Since the first form suffers from state explosion problem as the increase of the number of vulnerabilities, the second form has gained more popularity for its better scalability.

2.2. *Post-processing of attack graphs*

In order to make AG more useable, two ranking algorithms were designed for reducing complexity (Mehta et al., 2006; Sawilla and Ou, 2008), visualization techniques were applied to improve understandability (Lippmann et al., 2007; Homer et al., 2008), an incremental algorithms was designed for improving adaptability (Saha, 2008), and a number of systems and tools were developed, such as NetSPA (Ingols et al., 2009), CAULDRON (Jajodia and Noel, 2009), MulVAL (Ou et al., 2005). Our work does not focus on the specific generation of AG or system-level vulnerabilities, it rather slightly modifies dependency AG as a design basis of our holistic approach.

2.3. *AG-based risk assessment*

Using AG-based vulnerability analysis for risk assessment and proactive defense is a well-studied topic. In Noel et al. (2003), Wang et al. (2006), dependency AG was applied to compute minimum cost-hardening measures by identifying those key attack paths (AG edges). Then security metrics were introduced to AG for measuring network security risks using dynamic Bayesian network (Frigault et al., 2008), which was recently used to capture uncertainties implied by AG (Xie et al., 2010), such as uncertainties in attack structure, attacker action, and intrusion alerts, whereas the aforementioned work did not consider that. With the similar objective, our approach adopts HMM instead, because it allows us to introduce a quantitative model for specifying cost factors and to design heuristic algorithm for automated inference, in addition to capturing those uncertainties. We tackle the issue not only from an engineering perspective, but also take into account economic concerns.

To achieve optimal security hardening in enterprise networks, Dewri et al. formulated attack tree-based vulnerabilities patching as a multi-objective optimization problem and used genetic algorithm to search for the solution (Dewri et al., 2007). Our approach has fundamental difference because it uses dependency AG rather than attack tree (state-based AG) to represent network observations. Thus, thanks to HMM, the probabilistic mapping between AG-represented observations and estimated states may manifest those root causes of potential attacks, avoiding state explosion even in large scale networks.

3. Design motivation: an example scenario

We use an example network to illustrate the problem we intend to solve. The example will be also used for our experiments.

3.1. Network observations

The example is shown in Fig. 2, in which file server *FS* stores confidential information such as user profiles and business plans, database backend *DB* responds to external queries via web server WS. A legitimate user can login any hosts using three types of accounts: guest, legitimate user and root, while *DB*, *FS*, WS can only be maintained by a user with root privilege. We also assume a firewall *FW* to adopt loose filter policies and an intrusion detection system (IDS) to monitor network traffic. We further assume that the hosts have some initial vulnerabilities, which are summarized in Table 1 and indexed by CVE number.

In particular, *FS* is accessible to anonymous users, and it has vulnerabilities which allow `write` operation on its home directory and an executable command shell assigned to a FS user; H_w, WS and FS can be exploited for local buffer overflow via system vulnerabilities such as CVE-2007-0086[3], and H_a has a trust relationship with H_w, allowing H_a to have `write` operation on H_w. Formally, we introduce two grammars to describe those facts:

- *Com(x, y, p, u)*: host *y* can be connected by host *x* via protocol *p* with privilege *u*;
- *Atk(x, y, v)*: host *y* can be compromised by *x* by exploiting vulnerability or launching attack *v*.

For example, *Com*(H_w, *WS, http, user*) implies that H_w can access web server *WS* via protocol http as a common user; *Atk*(H_a, *FS, BoF*) represents that H_a may compromise file server FS by buffer overflow attack.

[3] CVE – Common Vulnerabilities and Exposures, and the details of the numbered vulnerabilities in this article can be referred to http://nvd.nist.gov/.

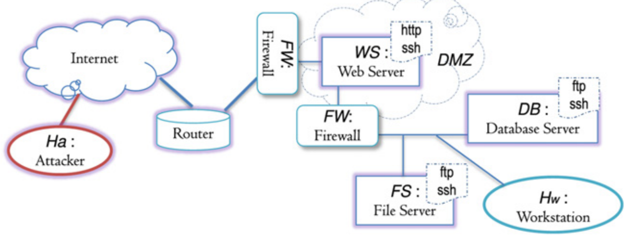

Fig. 2 – A network example.

3.2. Potential attacks

In this example network, we hypothesize that a sophisticated attacker intends to deface web server WS, steal sensitive data from file server *FS*, or crack database server *DB*. In particular, the attacker can do the following to deface web server:

1. Examining the trust relationship between host H_a and H_w, and exploiting the vulnerabilities in H_w such as CVE-2007-5616 to obtain root privilege, i.e., *Atk*(H_a, H_w, *U2R*);
2. Remotely accessing WS with root privilege of H_w and defacing the webpage, i.e., *Atk*(H_a, *WS, Defacing*).

The attacker may alternatively exploit vulnerability CVE-2008-3257 in WS for privilege escalation. That says, even for the same attack goal, different attackers may take different attack paths. Nevertheless, a preliminary yet essential step is to examine system vulnerabilities, especially those zero-day ones, as their patches usually take weeks or even months to be released, distributed and adopted even after they are identified. Such software vulnerabilities, however, does not undermine the significance of other vulnerable configurations, such as *Com*(H_a, WS, *ftp, guest*) and *Com*(H_a, H_w, *ssh, user*), which may serve as stepping stones for attacks.

3.3. Security hardening

In practice, it is neither meaningful nor feasible to exhaustively search and patch all the identified vulnerabilities, especially in large-scale computer networks. A more practical way for a SA is to selectively deal with the most significant vulnerabilities (or root causes)[4], by balancing the potential loss due to attack and defense. In this example, we may refer to the definitions in ICAT metabase (NIST) and specify a set of metrics to quantify attack consequence and defense cost. For instance, attack consequence can be measured by confidentiality loss, denial of service, public embarrassment, privilege escalation, integrity loss, while defense cost factors may involve system downtime, installation cost, operation cost, training cost, incompatibility cost, and so on. Furthermore, the SA has to specify a pool of countermeasures as defense strategies. For simplicity, we define four basic operations for the example network:

[4] We define *root causes* as a number of vulnerabilities which play important roles in an attack, and they may vary with attack targets. That says, root causes of attack A are not necessarily root causes of attack B.

Table 1 – Initial vulnerabilities in example network.

Host	CVE#	Protocol	Vulnerability	CVSS severity
H_w	CVE-2008-3234	ssh	Remote-2-User (R2U)	6.5 (Medium)
	CVE-2007-5616	ssh	User-2-Root (U2R)	7.2 (High)
FS	CVE-2001-0755	ftp	Buffer overflow (BoF)	7.5 (High)
	CVE-2006-2421	ssh	Buffer overflow (BoF)	7.5 (High)
WS	CVE-2008-2384	http	SQL injection	7.5 (High)
	CVE-2007-6388	http	XSS vulnerability	4.3 (Medium)
	CVE-2007-0086	http	DoS	7.8 (High)
	CVE-2006-3747	http	DoS	7.6 (High)
DB	CVE-2007-6304	SQL	DoS	5.0 (Medium)
	CVE-2001-1180	setuid	User-2-Root (U2R)	7.2 (High)

- *Disconnect(i,j)*, where $i \in \{H_w, WS, FS, DB\}$, $j \in \{Internet, LAN\}$, in total 8;
- *Patch(i,j)*, where *i* is any host, *j* is system vulnerability, in total 12;
- *Disable(i,j)*, where *i* is any host, $j \in \{ssh, ftp, http\}$, in total 12;
- *Configure(i)*, where $i \in \{H_w, FW, IDS\}$, in total 3.

Thus, we obtain 35 countermeasures in total. Although the countermeasure set is small, an exhaustive combination between them is impractical for obtaining cost-effective security policies, since the searching space would become prohibitively large as the number of countermeasures increases and the size of network grows. We are therefore motivated to develop such an approach which can (1) identify the root-cause vulnerabilities in a given network; (2) infer the most cost-effective security hardening given predefined security metrics and cost factors. More importantly, the approach must be automated, scalable and adaptable so as to dynamically accommodate the situations where the amount of network assets and vulnerabilities become increasingly large.

4. Design rationale: a probabilistic rather than deterministic approach

AG-based vulnerability analysis is essential to risk assessment and proactive response in enterprise networks, but merging the gap between low-level system events and high-level organization concerns remains as a challenge. Following the design outline shown in Fig. 1, our approach systematically tackle it through four major steps,

1. To abstract and represent the network observations of interest;
2. To explore the probabilistic mapping between explicit observations and implicit system states for capturing relevant uncertainties;
3. To identify and associate system states with prescribed security metrics;
4. To infer cost-effective security hardening through benefit-cost analysis.

A more detailed operational context and working flow of AG-HMM is shown in Fig. 3, starting with the collected network observations of interest (input) that will be represented by dependency AG and ending with a set of optimally selected countermeasures (output). The operation is a recursive refinement process. Please note that the automated generation of Attack Graphs is out of the scope of this article, and we simply assume that they have been generated in some tools like MulVAL (Ou et al., 2005).

4.1. *Using dependency attack graph to represent observations*

We use *observation* to refer to the observable subjects that are used to characterize attack states. Since the observations are numerous, complex and inherently interleaved, spanning hardware and software components across the network, the computing infrastructure, and human relationships (Bahl et al., 2007), we use attack graph, a typical vulnerability analysis method, to represent the observations associated with attacks. In particular, we slightly modify multiple-prerequisite graph (MP) (Ingols et al., 2009) by adding three types of labels to the observations:

Solid observation Z_1: physical components and various network assets, including hosts (servers, workstations) with IP addresses, gateway, router, host connectivity;
Soft observation Z_2: the subjects which are observable and measurable, such as network traffic, software application/service, trust relation, user privilege;
Dark observation Z_3: system vulnerabilities and other anomalous events.

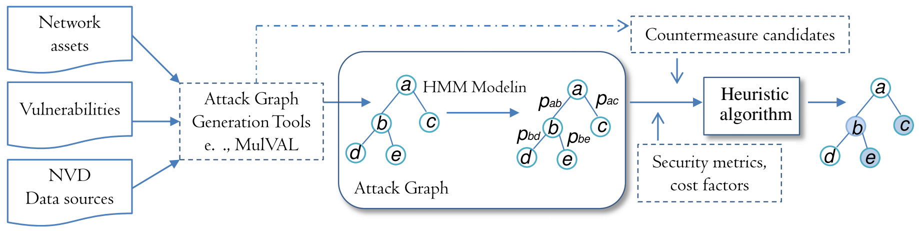

Fig. 3 – Working flow of AG-HMM. We assume that some appropriate tools like MulVAL can be used to generate attack graphs; the purpose of HMM is assign probabilities to graph edges, which are then processed by a heuristic algorithm by taking into account countermeasure candidates and predefine cost factors. The shaded nodes represent root-cause vulnerabilities.

In doing so, we may obtain a more concise graph structure with less vertices and edges, while the original semantics remain unchanged. This classification is essentially motivated by the fact that different observations may play different roles in an attack, so they should be given different weights for estimating attack state. For example, the web deface attack described in Section 3.2 can be represented by our modified AG in Fig. 4.

4.2. *Using Hidden Markov Model to estimate attack states*

We introduce *attack state* to characterize attack consequence in terms of the loss of security properties such as availability, integrity, confidentiality, and it can be estimated by observations. We assume that attack states can be treated discretely and mapped with corresponding observations with certain probabilities, e.g., $P(S_i|O_i) = p_i$. This is because that, (1) neither attacker nor SA can directly monitor attack states by collecting all the observations at each point of time; (2) there are uncertainties between observations and actual attack states. For better illustration, we exemplify attack scenario shown in Fig. 4 and depict it in Fig. 5(a) (the corresponding nodes are defined in Table 2), which has the following implications:

- Observation set is dynamic, updating with system states;
- It is observation set O_i, which contains subsets O_i^1, O_i^2, O_i^3 that respectively belong to Z_1, Z_2, Z_3, rather than individual observation v_x that determines state estimate;
- For each observation set, there are a number of key ones (root causes) which play more significant roles than the others in estimating state.

For example, in Fig. 5(a), as a result of exploiting prerequisites $O_1 = \{v_1, v_2, v_4, v_5\}(v_1, v_2 \in O_1^1 \subset Z_1,\ v_4, v_5 \in O_1^2 \subset Z_2)$, network transits from state S_0 (normal state) to S_1, along with the occurrence of new vulnerabilities $v_7, v_8 (v_7, v_8 \in O_3^3 \subset Z_3)$. This process continues until the target state S_N (S_3 in this example) is achieved, along with observation O_N. Fig. 5(b) illustrates a general case: prerequisite observations v_5, v_1, v_y contributes to state $\S_i$ with probabilities p_{5i}, p_{1i}, p_{yi} respectively. As a result of state transition from S_i to S_j, observation v_x is

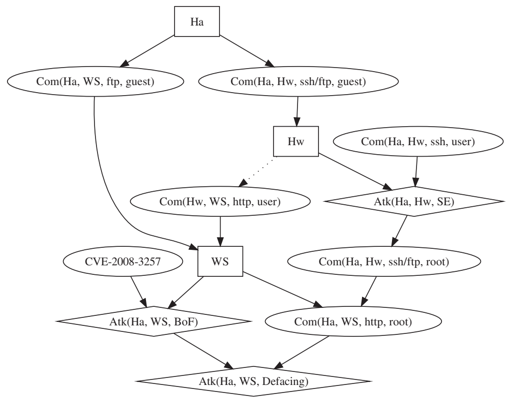

Fig. 4 – Defacing web server WS. Shown are solid observations (block), soft observations (ellipse), and dark observations (diamond).

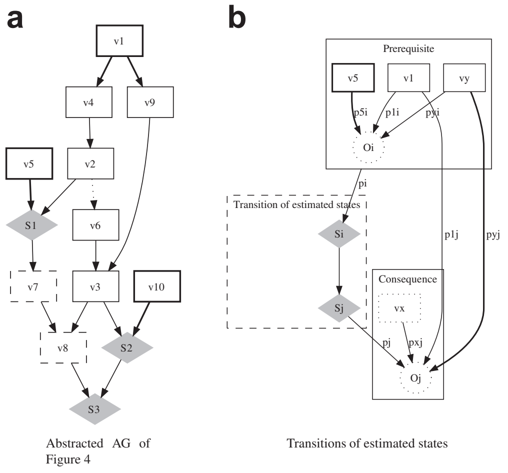

Fig. 5 – The probabilistic mapping between network observations and attack states. Shown are initial observations (solid block), root causes (thick block with outgoing arrow), new observations (dashed block), and virtual variable (dotted circle).

newly generated, and it can be used together with existing observations v_1 and v_y to estimate state S_j.

To explore the probabilistic mapping between network observations and states, we use Hidden Markov Model (HMM) to model and capture the relevant uncertainties. More formally, an HMM is featured by a triple tuple $\lambda = (A, B, \pi)$ (a more detailed tutorial is given in Rabiner (1989)),

- $A = \{a_{ij}\}$: state transition probability distribution, where $a_{ij} = P[q_{t+1} = S_j|q_t = S_i], \quad 1 \le i,\ j \le N$ (N is the number of states in the model);
- $B = b_j(k)$: the observation symbol probability distribution in state j, where $b_j(k) = P[v_k \text{ at } t|q_t = S_j], \quad 1 \le j \le N,\ 1 \le k \le M$ (where M is the number of observation elements per state);
- $\pi = \{\pi_i\}$: the initial state distribution, and $\pi_i = P[q_1 = S_i]$.

The construction of an HMM relies on the specification of states S, observations O, and three probabilities A, B, π. An instantiated model enables two operations,

- System states and their transitions $S_i \rightarrow S_j$ can be specified and associated with certain cost factors, providing a formal way to integrate security metrics;

Table 2 – Notations of graph nodes in Fig. 5(a).

Node	Notation	Node	Notation
v_1	H_a	v_8	$Com(H_a, WS, http, root)$
v_2	H_w	v_9	$Com(H_a, WS, ftp, guest)$
v_3	WS	v_{10}	CVE-2008-3257
v_4	$Com(H_a, H_w, ssh/ftp, guest)$	S_1	$Atk(H_a, H_w, SE)$
v_5	$Com(H_a, H_w, ssh, user)$	S_2	$Atk(H_a, WS, BoF)$
v_6	$Com(H_a, H_w, http, user)$	S_3	$Atk(H_a, WS, Defacing)$
v_7	$Com(H_a, H_w, ssh/ftp, root)$		

- It can either characterize a host or a network according to the granularity of the observations.

Based on the model, our approach AG-HMM aims at solving such a problem: Given a sequence of observation set $O = O_1O_2\ldots O_t$ (O_i contains a set of observable variables, and t denotes the discrete time) and the model λ, how to estimate state transitions $S = S_1S_2\ldots S_t$. This is significant for both attacker and SA: for an attacker, he/she may estimate whether the next state is the target; for an SA, she/he may decide whether some observations need to be removed to make next states more secure.

4.3. *Cost-benefit analysis for risk assessment*

It has shown that the goal-directed attacker behavior can be specified in terms of system state transitions, and an attack scheme exploiting a sequence of vulnerabilities can be viewed as state transitions from S_0 to S_N (Arnes et al., 2006; Zhang et al., 2009). By slightly modifying dependency AG, we model the exploit-dependence state transition as *directed state contact* graph $G = <S, E>$, where a set of vertices S represents system states, $E = O \times U$ are edges (O is a set of observations, and U is a set of actions[5]). Each directed edge represents a transition between two system states $S_i, S_j \in S$, and is denoted as a tuple $E_{ij} = <S_i, S_j, U_i, O_i>$, where S_i is the current state and S_j is the next state with $U_i \times O_i$, i.e., edge E_{ij} represents state transition $S_i \rightarrow S_j$. Thus attacker behavior can be described in the following way:

Definition 1. An attacker e's attack trace can be viewed as a sequence of action-dependence edges connecting initial and target states, i.e., $\tau(e) = (E_{01}, E_{1k}, \ldots E_{ij}, E_{jN})$.

The definition also implies that attackers sharing the same goal may take different actions even in the presence of same observations because of different attack capability. From the perspective of SA, it is more meaningful to identify and remove the root causes of attack for cost-effective security hardening. To do that, the optimal attacker behavior can be defined as follows:

Definition 2. In a directed system state contact graph G, an attacker e always intends to find the least-weight-path $E_{\min}$ from source node S_0 to destination node S_N.

This definition suggests an SA to seek the least-weight-path $E_{\min}$ in graph G and then remove the corresponding observations in O which serve as root causes.

Furthermore, the elements O, S of graph G can be provided by HMM, and the edge E_{ij} can be quantified using a cost factor C_{ij}, which is composed of C_{ij}^f and C_{ij}^m, representing attack cost caused by potential vulnerabilities and defense cost associated with security hardening respectively. That is, $C_{ij} = C_{ij}^f + C_{ij}^m$. To enable an SA to tune the trade-off between attack cost and defense cost according to specific application scenarios and security demands, we introduce an impact factor $\alpha \in [0,1]$, we thus have $C_{ij} = \alpha C_{ij}^f + (1-\alpha)C_{ij}^m$. In particular, C_{ij}^m can be estimated by SAs in terms of maintenance effort like patching, network disconnection, security products installation/configuration (Butler, 2002; Dewri et al., 2007; Noel et al., 2003), represented as $\overrightarrow{D} = (D_i), \quad 1 \le i \le L'$; C_{ij}^f can be specified as attack consequence in terms of a set of security metrics such as integrity loss, privilege escalation, service failure (Arnes et al., 2006; Butler, 2002; Mell et al., 2006; Mu et al., 2008), denoted as $\overrightarrow{F} = (F_i), \quad 1 \le i \le L$. The unit of F_i and D_i can be hours, dollars, or Likert scales.

By slightly modifying Butler's definition in Butler (2002), we introduce threat index $I(S_j)$ to measure attack consequence for state S_j, i.e., $C_{ij}^f = I(S_j) - I(S_i)$, where $I(S_j)$ can be quantified in the following steps:

1. Specify $\overrightarrow{F}$ by identifying each F_{ij} (metric F_i of S_j).
2. Assess a single value function V_{ij} for each metric F_{ij}, where $V_{ij}(F_{ij})$[6] is defined as follows:

$$V_{ij}(F_{ij}) = \frac{F_{ij}}{\max\limits_{j} F_{ij}} \times 100, \quad 1 \le i \le L. \tag{1}$$

3. Calculate $I(S_j) = \sum_{i=1}^{L} W_i V_{ij}(F_{ij})$, where W_i is a preference factor for each metric pre-defined by an SA, ranking from 1 to 100, to reflect the relative concerns of security metrics in different scenarios. W_i can be also normalized as $\beta_i = W_i / \sum_{k=1}^{L} W_k$:

$$I(S_j) = \sum_{i=1}^{L} \beta_i V_{ij}(F_{ij}), \quad \beta_i \in [0,1]. \tag{2}$$

Similarly, we can quantify defense cost C_{ij}^m by calculating $M(S_i)$ and $M(S_j)$ respectively, where $M(S_j)$ is defense cost at system state S_j and can be calculated as $I(S_j)$ by specifying, valuing and normalizing a set of factors associated with maintenance, along with corresponding preference factors, namely $M(S_j) = \sum_{i=1}^{L'} \beta_i' V_{ij}(D_{ij})$. As a result, a cost function associated with system state transitions is obtained:

$$\begin{aligned} C &= \sum_{1 \le i \ne j \le N} a_{ij} C_{ij} \\ &= \sum_{1 \le i \ne j \le N} a_{ij} \left[\alpha\left(I(S_j) - I(S_i)\right) + (1-\alpha)\left(M(S_j) - M(S_i)\right)\right] \end{aligned} \tag{3}$$

Note it is reasonable to assume $I(S_j) \ge I(S_i)$ and $M(S_j) \ge M(S_i)$, since both attack cost and defense cost monotonously increase as an attack advances ($S_i \rightarrow S_j$). So seeking the least-weight-path $E_{\min}$ in graph G is transformed into an optimization problem about minimizing C. As indicated in (Dewri et al., 2007), the optimization problem is single-objective if α is predetermined, otherwise it is multi-objective. Additionally, security metric $\overrightarrow{F}$ is usually human- and organization-specific, varying with different application scenarios, business goals, and security demands. Thus, specifying appropriate granularity of security metric as particular system states and then determining the optimal security hardening make it not a well-posed optimization problem due to many practical constraints.

[5] Here an action implies that some observations are selected from O and then are exploited for reaching state S_i.

[6] It enables each metric F_{ij} with different unit measure to be normalized to be unit-less so that the values can be summed together under a single standard scale.

4.4. *Applying heuristic algorithm to infer optimal security hardening*

The solutions to HMM have been extensively discussed (Rabiner, 1989), and its application in AG-HMM has two phases: determining parameters A, B, π via training, and using the constructed model λ to estimate/predict system states. In fact, AG-HMM *does not* require accurate estimate of parameter set A, B, π, which can be simply populated by SA's experience, historic data, risk assessment tools like CVSS. Some algorithms like Baum–Welch method can also be used to refine the parameters through iterative learning. Our focus is to develop an algorithm to automatically infer cost-effective security hardening implied in Eq. (3) based on the parameterized AG-HMM.

Obviously the formulated optimization problem can only be solved by approximate algorithms. We thus develop a heuristic algorithm drawn from Ant Colony Optimization (ACO) algorithm family (Cordon et al., 2002; Dorigo et al., 1996), which basically contains 3 steps (we omit algorithm details due to space constraint):

1. Populate initial observation set O_0 by significant vulnerabilities $\{v_i, \dots v_m\}$;
2. Put ants on the selected vertices that contain the key vulnerabilities in graph G, and the ants trace back their *adjacent vertices* in probabilistic manners according to HMM-based probability distribution matrix B;
3. Rank the edges E_{ij} with the amount of pheromone left by the ants, manifesting the most significant *vertices*.

The above steps are iteratively carried out until the termination criterion is met, i.e., a certain amount of root causes are found or a pre-defined cost C is satisfied.

Clearly, the core element of this algorithm is *ant*, which constructs candidate traces (a complete trace is a solution) for the problem by independently and iteratively computation. A complete trace contains a collection of correlated observations with respect to the state transitions between S_1 and S_N, while an intermediate trace only contains parts of them. At each step, each ant k computes a set of feasible expansions to its current trace and moves to one of these probabilistically according to a probability distribution p^k_{xy} (ant k from observation x to the next one y) by combining and specifying two parameters: ε_{xy}, the attractiveness of the move, as computed by some *a priori* desirability of that move; τ_{xy}, the trail level of the move, indicating how proficient is has been in the past to make that particular move, which is *a posteriori* indication of the desirability of that move.

Taking into account our specific concerns, and based on *a prior* knowledge that $b_j(x) = \Pr\{x|s_j\}$ and $b_j(y) = \Pr\{y|s_j\}$, we define ε_{xy} as the correlation coefficient of these two probabilities, $\varepsilon_{xy} = \mathrm{Cov}(b_j(x), b_j(y))/\sqrt{D(b_j(x))}\sqrt{D(b_j(y))}$, where $\mathrm{Cov}(b_j(x), b_j(y))$ is the covariance of $b_j(x)$, $b_j(y)$ and $\sqrt{D(b_j(x))}$ is the variance of $b_j(x)$. τ_{xy} can be calculated as $\tau_{xy} = \rho\tau_{xy} + \tau_0(1 - C_i/\overline{C})$, where $\rho \in (0,1]$ is a coefficient, and $1 - \rho$ represents the decrease of trail between two generations of complete traces, τ_0 is the initial trail level which is usually fixed to be an arbitrary small positive value, $\overline{C}$ is a moving average on the cost of the last l traces, and C_i is the cost of the ith ant's trace. Apparently the cost for state transition which generates a is taken as a hidden guidance for ant's movement. The probability for ant k to make a move is thus calculated as follows:

$$p^k_{xy,b} = \begin{cases} \dfrac{\tau_{xy}\cdot\xi + \varepsilon_{xy}\cdot(1-\xi)}{\sum\limits_{g\in tabu_k} \tau_{xg}\cdot\xi + \varepsilon_{xg}\cdot(1-\xi)}, & \text{if } y \notin tabu_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where the sum is overall the feasible moves and $\xi \in (0,1]$ is a control parameter balancing the relative importance of the trail τ_{ab} and the attractiveness ε_{ab}, $tabu_k$ is an observation dependent set for kth ant's feasible moves, and then p^k_{xy} is in fact a trade-off between the desirability of the current move and the past traces.

5. Experiments

Taking the example network given in Section 3 as a scenario, together with some realistic assumptions, we conduct a set of experiments to evaluate the feasibility and effectiveness of our approach. We use this scenario because the similar version is commonly used in AG research community (Jha et al., 2002; Sheyner et al., 2002; Sawilla and Ou, 2008; Mehta et al., 2006).

5.1. *Experiment settings*

Initially, a dependency AG should be generated in terms of network observations as discussed in Section 3, and a HMM model should be instantiated in the meantime using a historic dataset. Since our focus is on evaluating the performance of AG-HMM rather than AG generation, we manually generated a dependency AG-like graph and specified its vertices and edges as HMM-based elements and parameters, including coarse-grained observations O, system states S, and parameter set $\lambda = (A, B, \pi)$. In particular, the incoming edges of a node introduce preconditions and the outgoing edges direct to the consequence, each edge is given a value (corresponds to $b_j(k)$) as the success likelihood of generating an attack state.

In total we specify 14 system states (1 normal plus 13 anomalous) with initial occurring probabilities π_i. In particular, we assign the edge values associated with system vulnerabilities using the *exploitability index* given in NVD (NIST) (a detailed explanation can be found in Mell et al. (2006)), while the other values are scored and averaged by five experienced SAs. The initial state transitions are simply defined as follows:

$$a_{ij} = \begin{cases} 0.9, & \text{if } i = j \\ \dfrac{1}{130}, & \text{otherwise} \end{cases} \tag{5}$$

We also assume a sophisticated attacker who targets on three malicious goals:

Goal 1 (S_{11}) – defacing web server *WS*;
Goal 2 (S_{12}) – stealing sensitive data from file server *FS*;
Goal 3 (S_{13}) – cracking database server *DB*.

It is obvious that the number of overall network states of this example may easily achieve billions, as any update of observations (e.g., traffic patterns, protocols, services, user privileges) may result in state transitions. Thus, insisting a set of S that comprehensively reflects the true network state is neither meaningful nor practical. We instead only select a set of coarse-grained ones for examination. In addition, following the analysis in Section 4.3, a set of security metrics and cost factors is specified in Table 3 for measuring attack consequence (attack cost) and calculating defense cost.

Our implementation of AG-HMM mainly uses HMM toolbox for Matlab (version R2009a) on a PC with 2.8 GHz Intel(R) Core(TM)2 Duo CPU with 4G RAM running Windows Visa Business Service Pack 2. The performed experiments include: (1) seeking the root causes of the three attack schemes; (2) inferring optimal security hardening, and; (3) evaluating usability, adaptability, and scalability. Our algorithm was run 30 times with $\rho \in [0.01,0.1]$ for each experiment for compensating randomness.

5.2. *Pinpointing the root causes*

Since each malicious goal undergoes a number of stepping stones (corresponding to implicit state transitions), our first objective is to identify the *root causes*, namely the most significant observations. As discussed in Section 3.2, the attack generally contains three types of step stones: remote-to-user (either detectable or stealthy), remote-to-root (either detectable or stealthy), and user-to-root (stealthy). Apparently a sophisticated attacker prefers stealthy attacks.

We specified cost C in Eq. (3) as attack cost C^f in terms of the security metrics defined in Table 3, and then introduced it to the algorithm parameter τ_{xy}. The system states in terms of C^f in this experimental scenario is thus calculated as Eq. (2) and shown in Fig. 6. For example, at sate S_5, the scale of *Public embarrassment* is given 1, and *Integrity loss* is set as 3-scale, leading to a total attack cost $1/2 \times 100 \times 0.20 + 3/6 \times 100 \times 0.05 = 12.5$.

Given the constructed HMM model, the algorithm converged rapidly to a stable state within 100 epochs. We omitted the minor results and visualized the most important portion in Fig. 7 via Graphviz (Graph Visualization Software), illustrating the state transitions along with the key observations. Specifically, we have the following findings:

Table 3 – Security metrics $\vec{F}$ and defense cost factors $\vec{D}$.

	Metric	Unit	Preference factor W_i	Weight β_i
$\vec{F}$	Confidentiality loss	0–6 Scale	100	0.33
	Denial of service	0–6 Scale	80	0.27
	Public embarrassment	0–2 Scale	60	0.20
	Privilege escalation	0–2 Scale	45	0.15
	Integrity loss	0–6 Scale	15	0.05
$\vec{D}$	System downtime	Hours	100	0.32
	Installation cost	Monetary	80	0.26
	Operation cost	Monetary	60	0.19
	Training cost	Monetary	40	0.13
	Incompatibility cost	0–2 Scale	30	0.10

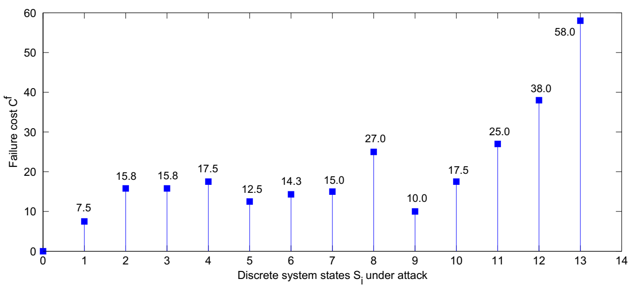

Fig. 6 – Attack cost C^f of estimated system states.

- S_1 serves as a *pivot* of attacks (detectable), which means that attacker gives this state the highest priority, and by achieving this state most of the ultimate malicious goals can be accomplished (the probability of state transition $S_1 \rightarrow S_4$ is higher than those of other states initiated from S_1). However, since S_1 can be easily detected, a sophisticated attacker may prefer the alternatives. This is confirmed by our observation that state transitions implicitly rely on cost factor C^f. For example, $S_0 \rightarrow S_6 \rightarrow S_{13}$ ($C^f_{6,13} = 43.7$) more likely occurs than $S_0 \rightarrow S_8$ ($C^f_{0,8} = 27.0$), even though the latter one is technically easier to achieve.
- S_4 is pinpointed as another key state in that it has several vulnerabilities which can be easily exploited, and File Server (FS) stores user profiles of the whole network. The three malicious goals can be easily accomplished by achieving S_4. Clearly, both CVE-2001-0755 and CVE-2006-2421 play significant roles in obtaining S_4.
- In addition to those soft and dark observations, it is found that solid observation WS serves as a root cause in most key state transitions in that its exploitation may easily lead to the state S_{11} (by XSS attack via CVE-2007-6388) and S_{13} (by SQL injection via CVE-2008-2384), two of the three malicious goals.

In order to validate the results obtained by AG-HMM, we randomly generated 10,000 observation probabilities by assigning a deviation range [0.01,0.10] and inferred the state transitions using viterbi algorithm. The results indicated that state sequence $S_0S_1S_4S_{12}S_{13}$ has the highest occurring probability, and the second one is $S_0S_1S_{10}S_{12}$, while the other 3 out of the top 5 ranked traces were same. This verifies the high *sensitivity* and *accuracy* of AG-HMM. Moreover, one may see that Fig. 7 can be viewed as a high-level representation of dependency AG, which is usually generated in terms of vulnerabilities with a complicated structure that overwhelms SA's comprehension. In particular, attack cost C^f potentially serves as an index to rank the AGs (the goal of work (Mehta et al., 2006; Sawilla and Ou, 2008)) and highlight those significant observations for a succinct view.

5.3. *Inferring cost-effective security hardening*

Given the *root causes*, a SA is expected to take the most effective security hardening by balancing defense cost and attack cost. In this experiment, we heuristically select a number of important countermeasures from a pool of

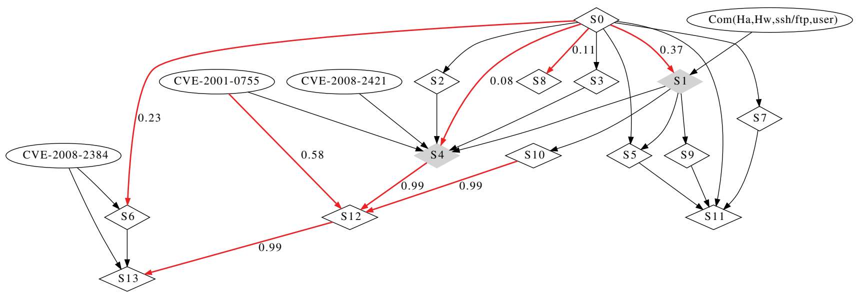

Fig. 7 – Inferred state transitions along with key observations. Shown are attack states (diamond), key states (filled diamond), major state transitions (red line with probabilities), and key observations (oval). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

candidates previously defined in Section 3.3, which are difficult to be leveraged by SAs. The defense cost C^m and attack cost C^f associated with each countermeasure are also calculated, which are shown in Table 4. In particular, as Butler (2002) points out that determining C^m needs a long historic data, which is not readily available for us. Thus, we simply transform the units of cost factors ($\overrightarrow{D}$ in Table 3) into unit-less scales and associate the values with the significance of network assets, where *DB* is assigned 1.0, *FS*, *WS*, H_w are set as 0.8, 0.5, 0.1 respectively. For example, *Disconnect*(*DB*, *Internet*) is calculated as $0.32 \times 100 \times 1.0 = 32$, while the defense cost of *Disconnect*(H_w, *Internet*) would be $0.32 \times 100 \times 0.1 = 3.2$. Attack cost is calculated in the same way.

Once root-cause vulnerabilities have been identified, a cost-effective security hardening can be inferred as follows,

- CM_{11} and CM_{12} can be taken to tighten access control over H_w, in worst case the trust relationship between H_w and H_a can be withdrawn;
- Vulnerability CVE-2008-2384, which makes *WS* vulnerable to SQL injection must be removed (CM_4), as it may directly lead to state S_{13} ($a_{0,6} = 0.23$);
- Vulnerability CVE-2001-0755 is highly suggested to be patched, which potentially allows H_a to conduct buffer overflow attack on the hosts running *ftp*, leading to states S_4, S_{10} and S_{12} with the probabilities 1.00, 1.00 and 0.72 respectively;
- Vulnerability CVE-2006-2421 should be also patched, since it may enable the attacker to gain root privilege on *FS* directly (leads to state S_4) via remote buffer overflow.

One may see that the inferred security policy aims to minimize the attack cost regardless of defense cost, implying a fact that all the suggested countermeasures should be taken to remove those stepping-stone attacks. However, such a perfect security policy is not always desirable in practice. For example, the trust relationship between H_w and H_a may not be easily terminated without sufficient evidence, and it is perhaps overlooked for a long time. Therefore, near-optimal solutions are more likely to be adopted by taking into account the cost factors we have analyzed.

To gain further insights into the cost-driven context, we replaced C^f in the last experiment as objective function Eq. (3) and reran the algorithm. Fig. 8 shows the results of the three attack goals respectively. Surprisingly, we observed that varying impact factor α (from 0.1 to 1.0 with step of 0.1) only had slight impact on the results, namely leading to the shift of total cost. This essentially indicates that the inferred solutions are *robust*. Further analysis manifests the fundamental of our design: a perfect security policy intends to remove all the possible observations regarding a key system state, while a cost-effective security policy simply reduces the probability $b_j(k)$ (success likelihood to reach state S_j) to be lower than a hidden threshold that is indirectly determined by α. Henceforth, $b_j(k)$ in this experiment did not undergo significant changes by adjusting α due to the small observation set associated with each network state we hypothesized. As a result, one cost-effective security hardening is obtained for each attack goal,

Goal 1: $CM_6 \wedge CM_{11},\ C^m = 14.9,\ C^f = 12.5$
Goal 2: $CM_7 \wedge CM_9 \wedge CM_{14},\ C^m = 17.5,\ C^f = 20.7;$
Goal 3: $CM_8 \wedge CM_{11},\ C^m = 14.3,\ C^f = 20.9.$

Table 4 – Selected countermeasures and their defense cost.

Notation	Countermeasure	C^m	Notation	Countermeasure	C^f
CM_1	*Disconnect*(*DB*, *Internet*)	32.0	CM_8	*Disable*(*DB*, *ssh/ftp*)	19.0
CM_2	*Disconnect*(*FS*, *Internet*)	25.6	CM_9	*Disable*(*FS*, *ssh/ftp*)	15.2
CM_3	*Disconnect*(*WS*, *Internet*)	16.0	CM_{10}	*Disable*(*WS*, *ftp*)	9.5
CM_4	*Patch*(*DB*, CVE-2008-2384)	26.0	CM_{11}	*Disable*(H_w, *all*)	1.9
CM_5	*Patch*(*FS*, CVE-2001-0755)	20.8	CM_{12}	*Configure*(*FW*)	29.0
CM_6	*Patch*(*WS*, CVE-2006-3747)	13.0	CM_{13}	*Configure*(*IDS*)	29.0
CM_7	*Patch*(H_w, *all*)	2.6	CM_{14}	*Configure*(H_w)	2.9

Disconnect(•) costs system downtime, *Patch*(•) incurs installation cost, *Disable*(•) causes operation cost, and *Configure*(•) results in both operation cost and incompatibility cost.

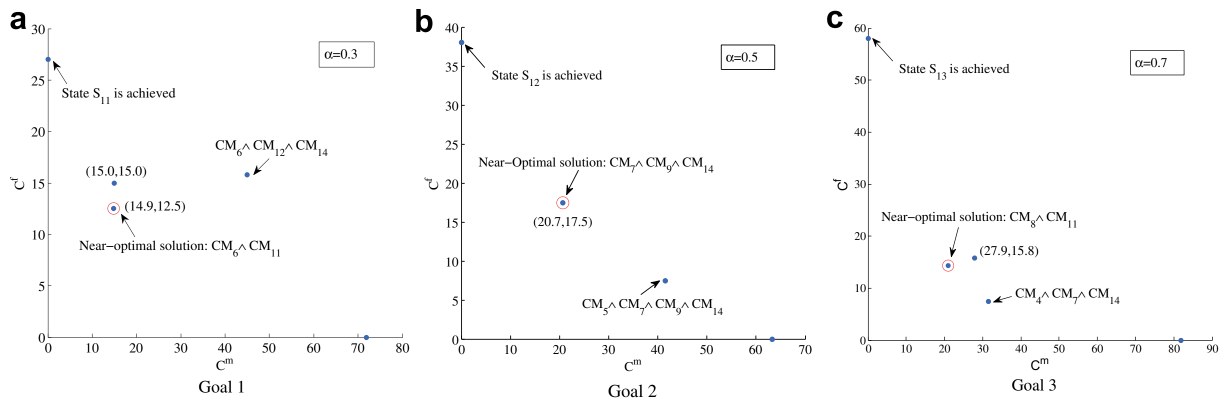

Fig. 8 – Solutions obtained by balancing the trade-off between defense cost and attack cost for each attack goal independently.

5.4. *Performance validation in a VoIP scenario*

The previous experiments have evaluated the *feasibility* and *effectiveness* of our approach. Despite the simple topology and the small amount of observations and states, the previous experiment shows how the AG-represented observations can be probabilistically correlated with attack states. In order to further evaluate our approach in terms of *usability*, *Sensibility* and *scalability*, we conducted an experiment in a more complicated VoIP network scenario, as shown in Fig. 9. We used this scenario because it was used in Howard et al. (2008) for evaluating an intrusion detector placement scheme, thus allowing us to compare the results.

5.4.1. *Settings*

This experiment is specifically set as follows,

- In addition to those original hypothesized vulnerabilities and attack traces, we assigned some initial probabilities A, B, π to construct an HMM and randomly generated 20,000 observation samples for its training.

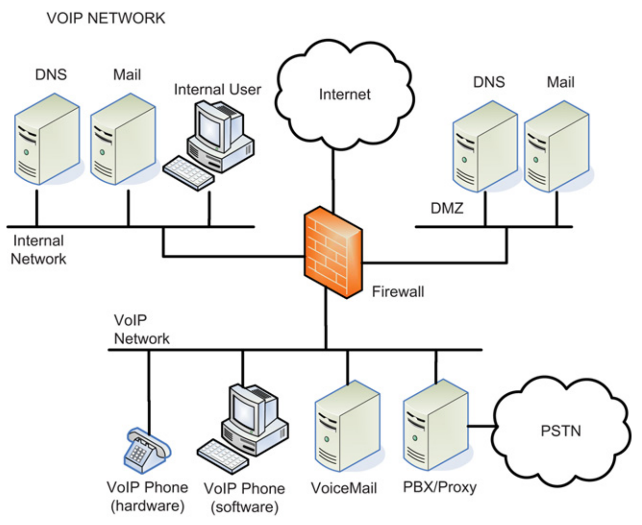

Fig. 9 – A voice over IP network.

- Taking into account the significance of network assets, the metrics in Table 3 were specified as *Confidentiality* (100), *Privilege escalation* (50), *DoS* (30), *Integrity loss* (20), and *Public embarrassment* (10).

5.4.2. *Results and analysis*

For easier illustration, the results are represented as an attack graph and shown in Fig. 10, where S_8, S_{10} and S_{11} were flagged as key states. To make the graph succinct, we omit state transition probabilities and relevant observations, explicitly denoting attack states. The key state transition sequence is highlighted in red and detailed as follows.

S_1: $Atk(H_a, DNS, Ping/Traceroute) \rightarrow$
S_2: $Atk(H_a, DNS, PortScan) \rightarrow$
S_3: $Atk(H_a, MailServer, Ftp\_BoF) \rightarrow$
S_6: $Atk(H_a, DNS, Download\_Sniffer) \rightarrow$
S_8: $Atk(DNS, DNS, Sniffer\_Traffic) \rightarrow$
S_{11}: $Atk(H_a, VoicemailServer, SQL\_BoF) \rightarrow$
S_{14}: $Atk(VoicemailServer, PSTN, ARP\_Poisoning) \rightarrow$
S_{18}: $Atk(Host\_Win2K\_Softphone, VoIPNetwork, Eavesdrop)$

We observed that the results verify the deployment strategies of IDS sensors reported in Howard et al. (2008), and the state chain (together with another 3 with higher transition probabilities) clearly suggests a set of security hardening for the vulnerabilities associated with those key attack states. In particular, incoming traffic of *DNS server* should be carefully monitored, and an encryption scheme is desirable. *Vociemail server* is also identified as a key stepping stone, so its vulnerabilities with (IIS,SQL) requires immediate patches.

5.4.3. *Discussion*

5.4.3.1. *Usability*. Our experiments show that our approach is *user-centric*. In particular, we observed that cost function Eq. (3) directly impacts on the results for an initialized HMM instance, so an SA is able to specify different security metrics and preference factors (as shown in Table 3) according to the particular situation. In addition, the SA can further leverage attack cost and defense cost via impact factor α in Eq. (1) by taking into account organization-level factors. Clearly, in

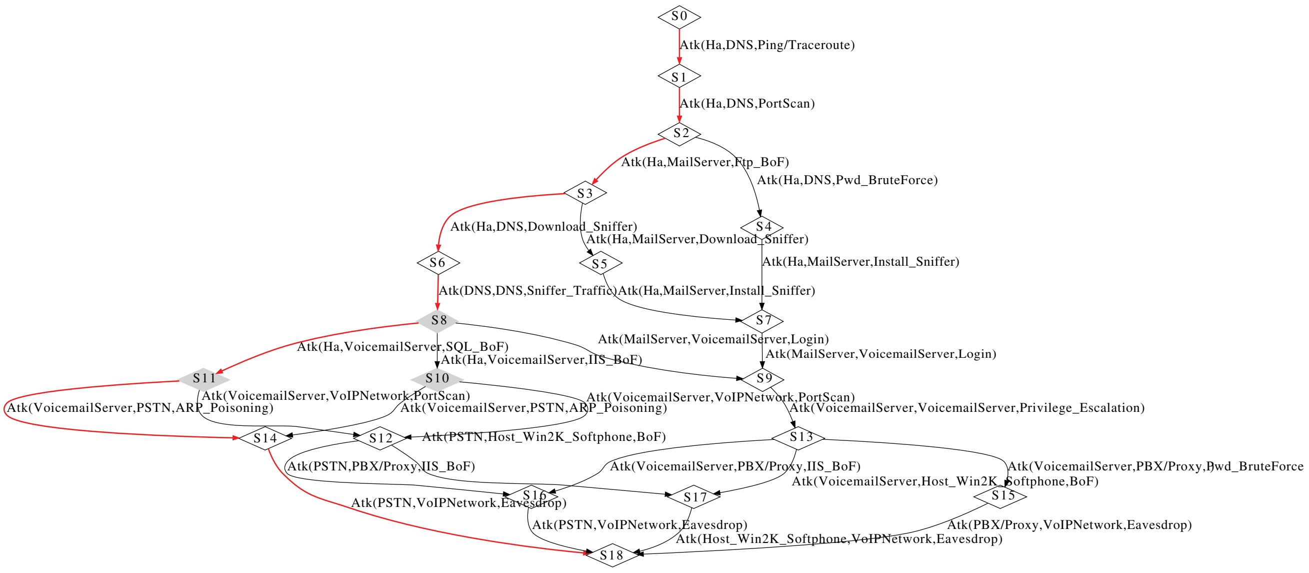
S0
Atk(Ha,DNS,Ping/Traceroute)
S1
Atk(Ha,DNS,PortScan)
S2
Atk(Ha,MailServer,Ftp_BoF)
Atk(Ha,DNS,Pwd_BruteForce)
S3
S4
Atk(Ha,DNS,Download_Sniffer)
Atk(Ha,MailServer,Download_Sniffer)
S6
S5
Atk(Ha,MailServer,Install_Sniffer)
Atk(DNS,DNS,Sniffer_Traffic)
Atk(Ha,MailServer,Install_Sniffer)
S8
S7
Atk(MailServer,VoicemailServer,Login)
Atk(Ha,VoicemailServer,SQL_BoF)
Atk(Ha,VoicemailServer,IIS_BoF)
Atk(MailServer,VoicemailServer,Login)
S11
S10
S9
Atk(VoicemailServer,VoIPNetwork,PortScan)
Atk(VoicemailServer,PSTN,ARP_Poisoning)
Atk(VoicemailServer,PSTN,ARP_Poisoning)
Atk(VoicemailServer,VoIPNetwork,PortScan)
Atk(VoicemailServer,VoicemailServer,Privilege_Escalation)
S14
S12
Atk(PSTN,Host_Win2K_Softphone,BoF)
S13
Atk(PSTN,PBX/Proxy,IIS_BoF)
Atk(VoicemailServer,PBX/Proxy,IIS_BoF)
Atk(VoicemailServer,PBX/Proxy,Pwd_BruteForce
Atk(VoicemailServer,Host_Win2K_Softphone,BoF)
S16
S17
S15
Atk(PSTN,VoIPNetwork,Eavesdrop)
Atk(PSTN,VoIPNetwork,Eavesdrop)
Atk(PBX/Proxy,VoIPNetwork,Eavesdrop)
Atk(Host_Win2K_Softphone,VoIPNetwork,Eavesdrop)
S18

Fig. 10 – State transitions (edges in red show the state transition sequence with key states in gray). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

addition to those scenario-specific parameters, an SA is not required to configure AG-HMM in details.

5.4.3.2. *Sensitivity*. In order to examine the *robustness* of our approach in the presence of varying parameters, we need to conduct sensitivity analysis. Basically, the model-based parameters can be classified as two external and internal ones: the parameters relevant to cost function are external ones, which can be specified by SAs; those initial parameters such as the occurrence probability of certain observations are internal ones. Also, while AG-HMM has two functional components, i.e., attack graph and HMM, our sensibility analysis is focused on the latter part. In particular, we varied the initial parameters of HMM within a reasonable range, and then arbitrarily removed or added the observation samples with a percentage of ±30%. We got two findings: (1) as long as the significant observations, such as the vulnerabilities with high CVSS severity score, was given higher probability than the others, the results would remain to be same regardless of specific values; (2) the key state transition path would not be identified if any significant observation was missed. For instance, the state transition chain in the previous section would not be identified if *buffer overflow vulnerability* at Voicemail Server does not exist, which means state S_{11} would been never achieved.

5.4.3.3. *Scalability*. Compared to the previous experiment, the extra running time of our approach in this VoIP scenario, which has a larger state space (19 in total), is negligible, which respectively last 27.23*s* and 29.11*s*. We also observed that the heuristic algorithm generally converged fast regardless of the size of state space and observation space, demonstrating its potential scalability in large-scale networks and more complicated attack scenarios. In addition, a preliminary theoretic analysis shows that the upper bound of time complexity is $O(MN^2)$, where M is the number of network states, and N is the number of hosts, most of which is introduced by the generation of AG.

6. Conclusion

We developed an approach to bridge vulnerability analysis with risk assessment, with the ultimate objective to infer cost-effective security hardening in cost-aware enterprise networks. Our approach takes AG-represented network observations as input, and uses HMM to explore the probabilistic natures between explicit observations and implicit network states. It then employs a cost-driven heuristic algorithm to search for the optimal security hardening from a pool of countermeasure candidates. Our simulation-based experiments demonstrated that AG-HMM can significantly reduce the number of observations by associating them with relevant states, it meanwhile avoids state explosion suffered by many state-based security evaluation techniques.

REFERENCES

Ammann P, Wijesekera D, Kaushik S. Scalable, graph-based network vulnerability analysis. In: Proc. of CCS'02. p. 217–224.
Anderson R, Moore T. The economics of information security. Science Oct. 2006;314:610–3.
Arnes A, Valeur F, Vigna G, Kemmerer RA. Using hidden markov models to evaluate the risks of intrusions: system architecture and model validation. In: Proc. of RAID 2006. p. 145–164.
Bahl P, Chandra R, Greenberg A, Kandula S, Maltz DA, Zhang M. Towards highly reliable enterprise network services via inference of multi-level dependencies. In: Proc. of ACM SIGCOMM 2007. p. 13–24.
Butler SA. Security attribute evaluation method: a cost-benefit approach. In: Proc. of ACM ICSE'02. p. 232–240.

Cordon O, Herrera F, Stutzle T. A review on the ant colony optimization metaheuristic: basis, models and new trends. Mathware and Soft Computing 2002;9(2–3):141–75.

Dewri R, Poolsappasit N, Ray I, Whitley D. Optimal security hardening using multi-objective optimization on attack tree models of networks. In: Proc. of ACM CCS'07. p. 204–213.

Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B 1996;26:29–41.

Frigault M, Wang L, Singhal A, Jajodia S. Measuring network security using dynamic Bayesian network. In: Proc. of QoP'08. p. 23–29.

Graph Visualization Software, http://www.graphviz.org/.

HMM toolbox for Matlab, http://www.cs.ubc.ca/murphyk/Software.

Homer J, Varikuti A, Ou X, McQueen M. Improving attack graph visualization through data reduction and attack grouping. In: Proc. of IEEE VizSEC 2008. p. 68–79.

Ingols K, Chu M, Lippmann R, Webster S, Boyer S. Modeling modern network attacks and countermeasures using attack graphs. In: Proc. of ACSAC 2009.

Jajodia S, Noel S. Topological vulnerability analysis: a powerful new approach for network attack prevention, detection, and response. In: Algorithms, architectures and information systems security. Indian statistical institute platinum jubilee series; 2009. p. 285–305.

Jaquith A. Security metrics: replacing fear, uncertainty, and doubt. Addison-Wesley; 2007.

Jha S, Sheyner O, Wing J. Two formal analyses of attack graphs. In: Proc. of IEEE CSFW-15; 2002. p. 49–63.

Kheir N, Cuppens-Boulahia N, Cuppens F, Deba H. A service dependency model for cost-sensitive intrusion response. In: Proc. of ESORICS 2010. p. 626–642.

Lippmann R, Ingols KW. An annotated review of past papers on attack graphs. Technical report. MIT Lincoln Laboratory; Mar. 2005.

Lippmann R, Williams L, Ingols KW. An interactive attack graph cascade and reachability display. In: Proc. of IEEE VizSEC 2007. p. 221–236.

Howard GM, Bagchi S, Lebanon G. Determining placement of intrusion detectors for a distributed application through Bayesian network modeling. In: Proc. of RAID 2008. p. 271–290.

Mehta V, Bartzis C, Zhu H, Clarke E, Wing J. Ranking attack graphs. In: Proc. of RAID 2006. p. 127–144.

Mell P, Scarfone K, Romanosky S. Common vulnerability scoring system. IEEE Security & Privacy 2006;4(6):85–9.

Mu CP, Li XJ, Huang HK, Tian SF. Online risk assessment of intrusion scenarios using D-S evidence theory. In: Proc. of ESORICS 2008. p. 35–48.

NIST, ICAT metabase, http://nvd.nist.gov/.

Noel S, Jajodia S, O'Berry B, Jacobs M. Efficient minimum-cost network hardening via exploit dependency graphs. In: Proc. of IEEE ACSAC 2003. p. 86–95.

Ou X, Govindavajhala S, Appel AW. MulVAL: a logic-based network security analyzer. In: Proc. of USENIX security symposium 2005. p. 113–128.

Rabiner LR. A tutorial on hidden markov models and selected applications in speech recognition. In: Proc. of the IEEE, vol. 77; 1989. p. 257–86. No. 2.

Ray I, Poolsappasit N. Using attack trees to identify malicious attacks from authorized insiders. In: Proc. of ESORICS 2005.

Saha D. Extending logical attack graphs for efficient vulnerability analysis. In: Proc. of CCS'08. p. 63–73.

Sawilla RE, Ou X. Identifying critical attack assets in dependency attack graphs. In: Proc. of ESORICS 2008. p. 18–34.

Schneier B. Attack trees. Dr. Dobb's Journal 1999;24:12.

Sheyner O, Haines J, Jha S, Lippmann R, Wing JM. Automated generation and analysis of attack graphs. In: Proc. of IEEE S&P'02. p. 273–284.

Swiler L, Phillips C, Ellis D, Chakerian S. Computer-attack graph generation tool. In: Proc. of DARPA information survivability conference and exposition; 2001. p. 146–61.

Wang L, Noel S, Jajodia S. Minimum-cost network hardening using attack graphs. Computer Communications 2006;29(18):3812–24.

Xie P, Li JH, Ou X, Liu P, Levy R. Using Bayesian networks for cyber security analysis. In: Proc. of DSN 2010. p. 211–220.

Zhang Z, Ho P-H, He L. Measuring IDS-estimated attack impacts for rational incident response: a decision theoretic approach. Computers & Security 2009;28(7):605–14.

Zhang Z, Wang S. Boosting logical attack graph for efficient security control. In: Proc. of ARES; 2012. p. 218–23.

Shuzhen Wang is an Associate Professor at School of Computer Science in Xidian University, China. He received his Ph.D. degree from School of Electronical & Mechanical Engineering, M.S. and B.S. degrees from School of Economics & Management, Xidian University, in 2005, 2003, and 2000 respectively. He was a visiting scholar at the University of Adelaide, Australia from Sep.2009 to Sep. 2010. His research interests are machine learning, the internet of things, intelligent computing, and computer vision.

Zonghua Zhang is an Associate Professor at Institute Mines-Telecom/TELECOM Lille1 of France. Previously, he worked as an expert researcher at the Information Security Research Center of NICT, Japan. He also spent two years as a postdoctoral researcher at the University of Waterloo, Canada, then at INRIA, France after earning his Ph.D. degree in information science from Japan Advanced Institute of Science and Technology (JAIST) in March of 2006. His research is centered around security and privacy-driven decision-making problems in various computer and communication networks, with current focus on cost-effective security management, privacy-preserving network forensics and reputation systems.

Youki Kadobayashi is the Associate Professor at Internet Engineering Laboratory, Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Youki received his Ph.D. in Computer Science from Osaka University in 1997. After being a Research Associate and Lecturer for the Osaka University Supercomputer Center, he became an Associate Professor at Nara Institute of Science and Technology in 2000. Since 2009, he has also been serving as Associate Rapporteur of ITU-T Study Group 17 Question 4, where he has been working on cybersecurity standards. His research interests include cybersecurity protocols, secure overlay networks, web application security, and privacy-preserving application services.