Assessing Security Risk to a Network Using a Statistical Model of Attacker Community Competence

Tomas Olsson

Swedish Institute of Computer Science
P.O. Box 1263, SE-164 29 Kista, Sweden
tol@sics.se

Abstract. We propose a novel approach for statistical risk modeling of network attacks that lets an operator perform risk analysis using a data model and an impact model on top of an attack graph in combination with a statistical model of the attacker community exploitation skill. The data model describes how data flows between nodes in the network – how it is copied and processed by softwares and hosts – while the impact model models how exploitation of vulnerabilities affects the data flows with respect to the confidentiality, integrity and availability of the data. In addition, by assigning a loss value to a compromised data set, we can estimate the cost of a successful attack. The statistical model lets us incorporate real-time monitor data from a honeypot in the risk calculation. The exploitation skill distribution is inferred by first classifying each vulnerability into a required exploitation skill-level category, then mapping each skill-level into a distribution over the required exploitation skill, and last applying Bayesian inference over the attack data. The final security risk is thereafter computed by marginalizing over the exploitation skill.

1 Introduction

In order to manage the dynamic nature of the security of a network, where new nodes are added and new softwares are installed, operators regularly run scanning tools such as Nessus to discover the network topology and existing vulnerabilities [1]. However these tools do not put vulnerabilities into the context of how these vulnerabilities can be exploited to obtain illegal access to resources in the network. Likewise, they do not automatically determine the impact to the system or the potential risk.

As a consequence, a great number of automated approaches to security analysis have been proposed during the last decade [2]. Some of these automated approaches define security metrics over the paths of an attack graph [3,4,5,6,7], while others define a security metric in terms of security risk [8,9,10,11].

In this paper, we follow the second path by proposing a novel approach to compute the security risk. Risk is usually defined as the expected loss or as defined in [12]:

S. Qing, C.J. Mitchell, and G. Wang (Eds.): ICICS 2009, LNCS 5927, pp. 308–324, 2009.

> Risk is a function of the **likelihood of a given threat-source's exercising a particular potential vulnerability**, and the **resulting impact of that adverse event on the organization**.

Accordingly, a risk computation consists of two parts: a probability estimation of a successful attack and an impact estimation.

Most of the papers referred to above, use a simple model for estimating the impact to the network. Typically, they either do not use an explicit model, e.g. only probabilities, or sum the assigned weights of compromised nodes. As an example, in [13], the security of two networks, one patched and one unpatched, was compared by computing the number of hosts removed from the attack graph or by assigning a weight of importance to each host as well. This would be equivalent to estimating the impact by counting the number of compromised hosts in the attack graph or by computing the sum of their weights. In addition, many of the papers that use probabilistic models do not describe of how to estimate these probabilities. Therefore, in this work, we extend existing work with a more advanced network impact model and with a novel approach for inferring probabilities over attack graphs using a Bayesian approach assuming that a record of historical attack data from a honeypot is given [14].

The network impact model consists of a data model in combination with an impact model over an attack graph. The data model models how data is copied between softwares in a network, thereby taking into account the dependencies between different hosts and the dependencies between different data flows. The impact model describes what impact individual vulnerabilities have on the data sets with respect to confidentiality, integrity and availability as well as the impact propagation from compromised data sets. By assigning loss values to data sets, we can estimate the cost of a successful attack. Thus, in our work, an operator does not need to understand the importance of each node in a network as is the case in the papers referred to above.

The probability estimation of a successful attack is computed by first classifying each vulnerability into a required exploitation skill-level category that denotes how hard it is to successfully exploit the vulnerability. Then, based on historical attack data, we can create a statistical model of the exploitation skill of the attacker community using a Bayesian approach. Last, the final security risk is computed by marginalizing over the exploitation skill.

For our model, we follow the terminology of the multi-prerequisite attack graph (MP attack graph) presented in [13], but we introduce our own notation. The MP attack graph is fast to create, easy to understand and easy to work with; it uses very few model elements and have been shown to scale to large networks [13]. However, we are not bound to any specific attack graph as long as we can model the flow of data.

Notice however, that our model subsumes the original model such that our model can instantiate an equivalent recommendation algorithm for patching as in [13]. By removing all data flows, assigning all probabilities of successful attacks to the constant value 1.0 and putting data sets with a uniform loss value of 1 at each node, we can reproduce the original recommendation algorithm. By assigning different loss values to the different data sets, we also can reproduce the weighted version of the original algorithm. In addition, in contrast to the original model, our model makes it possible to apply partial patches that instead of removing a vulnerability, increase the required exploitation skill-level.

The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the impact of vulnerabilities to an IT-system [15]. We take advantage of the expert knowledge gathered through CVSS by using the impact elements from CVSS version 2.

A simple network from [13] that is shown in Fig. 1(a) illustrates what sort of problem we want to address with our approach. In this simple example, attacks start at host A with root access. All other hosts, but the firewall (FW), have a single vulnerability instance, e.g. v_B, that can be exploited remotely via one single open port. The firewall will accept that hosts C and D communicate with host E and will deny all other communication. In Fig. 1(b), we show the MP attack graph where the triangles indicate vulnerability nodes, circles indicate attack states and rectangles indicate what nodes can be reached from an attack state. The problem we are addressing is how to compare the security of different networks with respect to their vulnerabilities.

The rest of the paper is organized as follows. Sect. 2 introduces our formalization of the network model and the attack graph from [13]. Sect. 3 describes the network impact model. Sect. 4 presents the model for computing the probability that an attack will successfully exploit a vulnerability. Sect. 5 describes how to compute the security risk given a historical record of attack data. Sect 6 applies our framework to the simple network example described above. Sect. 7 compares related work with the proposed framework.

2 Introducing the MP Attack Graph Model

Our approach begins with constructing a model of the network containing elements such as a network topology, and traffic rules. Then, we define the MP attack graph. In the next sections, we follow the terminology of the NetSPA tool presented in [13], but we introduce our own notation.

Basic Network Model. A *network topology* consists of a set of *hosts* N and a set of *links* E. Each host $h \in N$ has a set of *interfaces* $i(h) \subseteq I$ where I is the set of all interfaces in the network. Each link $l \in E$ connects a set of interfaces $i(l) \subseteq I$. Each interface $i \in i(h)$ has a set of *ports* $p(h, i)$.

A traffic rule allows source and destination software instances to communicate via two interfaces of a host. Each host $h \in N$ has a set of *traffic rules* $r(h)$. By setting the second interface in a traffic rule equal to the first, we can allow traffic to only flow from or to the host of the rule.

MP Attack Graph. The MP attack graph consists of three types of nodes: *prerequisites*, *attack states* and *vulnerability instances*.

A *prerequisite* is the means needed to gain access to a vulnerability instance. We use Q to denote the set of all prerequisites in a network. For each prerequisite $q \in Q$, $v(q)$ denotes the set of (reachable) vulnerability instances of q.

An *attack state* a consists of a host $h \in N$ and an attack level $o \in \{root, user, dos, other\}$. We use A to denote the set of all attack states. For each $a \in A$, an attacker will, by obtaining attack state a, also obtain a set of prerequisites $q(a) \subseteq Q$ available at a.

A *vulnerability instance* is any means that an attacker can use to gain access to a system. We let V denote the set of all vulnerability instances. Each software instance $s \in S$ has a set of vulnerability instances $v(s) \subseteq V$. By successfully exploiting a vulnerability instance $v \in V$, an attacker will obtain a single attack state $a(v) \in A$.

3 Modeling the Impact to a Network Using Data Flows

After defining the basic network model and the attack graph, we define the data model and the impact model. These are our own contributions, except for some borrowed notions from [15]. A more elaborate description of the network impact model can be found in [16].

3.1 Defining the Data Model

A *data set instance* is a copy of any set of data that we protect with respect to the three security aspects – confidentiality, integrity and availability – that we denote c, i and a. D denotes the set of all data set instances.

A *data set instance location* consists of a data set instance $d \in D$ and a location identifier with a host $h \in N$, an interface $i \in i(h)$ of host h and a port $p \in p(h, i)$ of interface i.

All data set instances are assumed to be processed by *software instances*. S denote the set of all software instances. For each host $h \in N$, $s(h) \subseteq S$ denotes all software instances at h. For each software instance $s \in s(h)$, $store(s)$ denotes all data set instances stored at s and for each interface $i \in i(h)$, $inputs(s, i)$ and $outputs(s, i)$ denotes the set of data set locations from where s retrieves and produces data set instances via interface i respectively.

We let $depends(s, d)$ denote the set of data set instances used by s to produce data set d. Then, either, for each data set instance $d \in depends(s, d')$, $d \in store(s)$ or there is an interface $i \in i(h)$ such that d has a data set instance location in $inputs(s, i)$.

A *data flow* for data set instance $d \in D$ from a producer software instance $s_0 \in S$ to a retriever software instance $s_n \in S$ is a sequence of hosts $(h_0, h_1, h_2, \ldots, h_{n-1}, h_n)$ in N where: (a) $n > 0$, (b) $outputs(s_0, i_0)$ and *input*- $s(s_n,\ i_n)$ contain the same data set instance location with host h_0, interface i_0, port $p_0 \in p(h_0, i_0)$ and data set instance d, (c) there exists links with interfaces connecting all hosts in the sequence, and (d) there exists a traffic rule at each host allowing traffic from source host h_0 to destination host h_n using as source and destination the location identifiers of the producer and the retriever softwares respectively.

In addition, a data flow f is *active* if the data set instance of f is depending on other data set instances through the *depends* relation and then each of these data sets are either stored at the producer or retrieved such that there must exist an active data flow f' for the retrieved data set instance with the producer of f as the receiver of f'.

3.2 Defining the Impact Model

As a means to define the local impact, we define that a data set instance $d \in D$ is *accessible* to a software instance $s \in S$ if either $d \in \mathit{store}(s)$ (d is stored), there exists an active data flow for d with s as receiver (d is retrieved) or there exists an interface i such that d has a data set instance location in $\mathit{outputs}(s, i)$ while for each $d' \in \mathit{depends}(s, d)$, either $d' \in \mathit{store}(s, d)$ or there exists an active data flow for d' with s as receiver (d is produced). Similarly, a data set instance $d \in D$ is *accessible* to a host $h \in N$ if there exists a software instance $s' \in s(h)$ such that d is accessible to s' or there exists an active data flow for d with a sequence of hosts containing h.

Since we want to model the impact of a successful attack to the three security aspects, confidentiality, integrity and availability, we define that for each vulnerability instances $v \in V$ and for each security aspect $e \in \{c, i, a\}$, $\mathit{impact}(v, e) \in \{None, Partial, Complete\}$ denotes the local impact that vulnerability instance v has on security aspect e at a host h. We borrow the values $None, Partial$, $Complete$ for the three security aspects from CVSSv2 [15].

For our purpose, we interpret the impact values of CVSSv2 such that for each security aspect $e \in \{c, i, a\}$, an attacker will *locally violate* the security aspect e of, in the case of impact value: (a) $None$, no data set instances, (b) $Partial$, all data set instances *accessible* to the software instance with an exploited vulnerability instance, and (c) $Complete$, all data set instances *accessible* to the host of the software instance with the exploited vulnerability instance. Then, we define that for each data instance $d \in D$, for each security aspect $e \in \{c, i, a\}$, for each subset of exploited vulnerability instances $V_{exploit} \subseteq V$ and for each host $h \in N$, $locallyViolated(e, d, h, V_{exp})$ denotes that d was locally violated by an attacker at host h with respect to e given $V_{Exploit}$.

Now, we define predicates that infer violation of data set instances at the network level:

Loss of confidentiality. For each data set instance $d \in D$, for each host $h \in N$ and for each subset of vulnerability instances $V_{exp} \subseteq V$, $lossOfConf(d, h,$ $V_{exp})$ holds true if *locally-* $Violated($ c, d, h, $V_{exp})$ holds true.

Loss of integrity. For each data set instance $d \in D$, for each host $h \in N$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfInteg(d,$ h, $V_{exp})$ holds true if either (a) $locallyViolated(i,$ $d, h, V_{exp})$ holds true, or (b) there exists $s \in S$ such that $d' \in \mathit{depends}(s, d)$ and there exists an active data flow f for d' where the receiver is s and there exists a host h' in the sequence of hosts of f where $h' \neq h$ and $lossOfInteg(d'$, h', $V_{exp})$ holds true.

Loss of availability. For each data set instance $d \in D$, for each host $h \in N$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfAvail($ d, h, $V_{exp})$ holds true if either (a) $locallyViolated$ (a, d, h, V_{exp}) holds true, or (b) there exists $s \in S$ such that $d' \in depends(s, d)$ and for each active data flow f for d' where the receiver is s there exists a host h' in the sequence of hosts of f where $h' \neq h$ and $lossOfAvail(d', h', V_{exp})$ holds true.

Generic loss of security. For each security aspect $e \in \{c, i, a\}$, for each data set instance $d \in D$ and for each set of vulnerability instances $V_{exp} \subseteq V$, $lossOfSecurity(e, d, V_{exp})$ holds true if there exits $h \in N$ such that either $e = c$ and $lossOfConf(d, h, V_{exp})$ or $e = i$ and $lossOfInteg(d, h, V_{exp})$ or $e = a$ and $lossOfAvail(d, h, V_{exp})$ holds true.

Lastly, by assigning a *loss value*, we can estimate a cost of exploiting a vulnerability instance with a certain loss of security. For each data set instance $d \in D$, and for each security aspect $e \in \{c, i, a\}$, $cost(d, e)$ denotes the loss value of data set d with respect to e.

4 Modeling the Probability of a Successful Attack

In [13], the authors assume a worst case scenario for a successful attack: a single attacker, starting at an initial attack state, will be able to exploit every vulnerability instance in the MP attack graph. However, it is not a reasonable assumption, since attackers might have different exploitation skills and vulnerability instances might have different required exploitation skills. We propose a different scenario where an attacker will be able to *try* to exploit all vulnerability instances it has gained access to, but that the probability of successfully exploit a vulnerability instance depends on the *required exploitation skill-level* of the vulnerability instance and the *exploitation skill* of the attacker. Notice that we view each attacker as an instantiation of the wider attacker community, and thus, we do not model each attacker individually, but as a group.

For convenience, we have chosen to use four different exploitation skill-levels that we define in terms of a required exploitation skill. We map each exploitation skill-level into a probability distribution over the required exploitation skill. Similarly, we have chosen the exploitation skill to be in the arbitrary range [0,1].

In addition, we assume that an expert has assigned a exploitation skill-level to each vulnerability instance, for instance, by analyzing a freely available database such as NVD [17].

Required Exploitation Skill-Level. First we define the required exploitation skill-level. For each vulnerability instance $v \in V$, $z(v) \in \{Low, MediumLow, MediumHigh, High\}$ denotes the exploitation skill-level required by an attacker to successfully exploit v. To make formulas shorter, we sometimes use the following notation: $z_0 = Low$, $z_1 = MediumLow$, $z_2 = MediumHigh$ and $z_3 = High$. Thus, instead of typing $z(v) = Low$, we type $z(v) = z_0$ and so forth.

Successful Exploitation. Then, we define the successful exploitation probability. For each vulnerability instance $v \in V$, $k \in [0, 1]$ denotes the actual exploitation skill of an attacker, $k_v \in [0, 1]$ denotes the required exploitation to successfully exploit v, $p(k_v|m_z, \sigma_z)$ denotes the probability density function over k_v (where k_v is normally distributed with parameters m_z and σ_z and $z = z(v)$), and $\Phi(k, z)$ denotes the probability that an attacker with exploitation skill k will succeed to exploit v:

$$\forall i \in \{0, 1, 2, 3\}, m_{z_i} = \frac{i}{4} + \frac{1}{8} \text{ and } \sigma_{z_i} = \frac{1}{5}, \tag{1}$$

$$\Phi(k, z) \propto \int_k^1 p(k_v|m_z, \sigma_z) dk_v = erf\left(\frac{k - m_z}{\sigma_z\sqrt{2}}\right) - erf\left(\frac{-m_z}{\sigma_z\sqrt{2}}\right) \tag{2}$$

where erf is the error function [18].

5 Computing the Risk

In order to compute the risk to a network, without relying too much on expert knowledge, we use Bayesian statistical inference over a historical record of attack data.

To gather historical attack data for whether vulnerability instances were successfully exploited or not, we propose using a high-interactive honeypot. A honeypot is a controlled computer created to be vulnerable to attacks and therefore, any occurring traffic is suspicious [19]. Hence, it easy to detect attack attempts. If the nature of a set of vulnerability instances in the honeypot is known, it should also be possible to verify whether they where successfully exploited [20]. Of course, it is also possible to add other compromises investigated by the operator of the network.

Notice though, that to keep the probability estimations up-to-date, we must limit how old attack data we take into account. For instance, it might be realistic to only use the last six months of data, since older data might make the risk value obsolete.

Recall from Sect. 1 that risk is a function of an impact value and the probability of a successful attack. Therefore, we define the total risk to a network from attacks starting at $a_0 \in A$ in attack graph $\mathcal{A}$ during next s time slots as the expected loss over all data set instances:

$$risk(a_0, s, X, \mathcal{A}) = \sum_{d \in D, e \in \{c,i,a\}} cost(d, e) \cdot P_e(d|a_0, s, X, \mathcal{A}) \tag{3}$$

where $P_e(d|a_0, s, X, \mathcal{A})$ is the *probability of at least one successful attack* on data d given the *historical attack data* X, a_0 and s.

Historical Attack Data. First, we define the historical record of attack data X over a time period T where X contains $n(X)$ number of attacks and and T contains $n(T)$ number of time slots. For each attack data instance $x \in X$, $z(x) \in \{z_0, z_1, z_2, z_3\}$ denotes the required exploitation skill-level of the vulnerability instance during the attack, and $e(x) \in \{0, 1\}$ denotes whether the attack was successful ($e(x) = 1$) or not ($e(x) = 0$).

Posterior Distribution. Second, we derive the posterior distribution for the exploitation skill k and a parameter λ, which governs the number of expected attack instances, using Bayesian inference. The posterior distribution given X is:

$$P(k, \lambda | X) \propto P(X | k, \lambda) \cdot P(k) \cdot P(\lambda) \tag{4}$$

where $P(X|k,\lambda)$ denotes the probability that X was generated by an attacker community with exploitation skill k and parameter λ, while $P(k)$ and $P(\lambda)$ denotes the prior distributions for k and λ, respectively. We let $P(k) \propto 1$ (uniform distribution) and $P(\lambda) = Gamma(a, b)$ (Gamma distribution). The prior distributions describe what we know about the parameters prior to having the real data. Assuming independence between k and λ, we have:

$$P(X | k, \lambda) = P(X|k) \cdot P(n(X)|\lambda) \tag{5}$$

where $P(X|k)$ is the probability of X given k and $P(n(X)|\lambda)$ is the probability of $n(X)$ number of attack instances during time period T. For $P(X|k)$ we have:

$$P(X|k) \propto \prod_{x \in X} P(x|k) \tag{6}$$

where for each attack data instance $x \in X$, $P(x|k)$ denotes the probability distribution over x given exploitation skill $k \in [0, 1]$:

$$P(x|k) = (1 - \Phi(k, z))^{(1-e(x))} \cdot \Phi(k, z)^{e(x)} \text{ where } z = z(x) \tag{7}$$

such that

$$P(X|k) \propto \prod_{i=0}^{3} (1 - \Phi(k, z_i))^{e_{-z_i}} \cdot \Phi(k, z_i)^{e_{z_i}} \tag{8}$$

where e_{-z_i} is the total number of unsuccessful attacks and e_{z_i} is the number of successful attacks on vulnerability instances with required skill-level z_i. Then, for $P(n(X)|\lambda)$ we assume a Poisson distribution with mean value $n(T) \cdot \lambda$ such that:

$$P(n(X)|\lambda) = \frac{(n(T)\lambda)^{n(X)} e^{-n(T)\lambda}}{n(X)!}. \tag{9}$$

Probability of Success. Last, we derive the probability of at least one successful attack by marginalization. In case of attack graph $\mathcal{A}$ and attacks starting in attack state $a_0 \in A$, for each data set instances $d \in D$ and for each security aspects $e \in \{c, i, a\}$, $P_e(d|a_0, \mathcal{A}, k)$ denotes the probability that an attacker from an attacker community with exploitation skill $k \in [0, 1]$ will successfully exploit vulnerability instances in $\mathcal{A}$ such that d will be compromised with respect to e, and $P_e(d^n|a_0, \mathcal{A}, k)$ denotes the probability that at least one of n attack attempts succeeds:

$$P_e(d^n|a_0, \mathcal{A}, k) = 1 - (1 - P_e(d|a_0, \mathcal{A}, k))^n. \tag{10}$$

Then, we can compute the probability of at least one successful attack on d with respect to e, given X and within next s time slots, as:

$$
\begin{aligned}
&P_e(d|a_0, s, X, \mathcal{A}) \propto \\
&\int_0^1 \int_0^\infty \sum_{n=0}^{\infty} P_e(d^n|a_0, \mathcal{A}, k)\ P(n(s)|\lambda)\ P(X|k, \lambda)\ p(k)\ p(\lambda) d\lambda dk = \\
&C - \int_0^1 \frac{1}{\left(1 + \frac{sP_e(d|a_0, \mathcal{A}, k))}{b+n(T)}\right)^{a+n(X)}} \prod_{i=0}^{3} (1 - \Phi(k, z_i))^{e_{-z_i}}\ \Phi(k, z_i)^{e_{z_i}} dk
\end{aligned}
\tag{11}
$$

where $n(s)$ is the number of attacks during next s time slots, for which we use a Poisson distribution with mean $s \cdot \lambda$, and C is a constant.

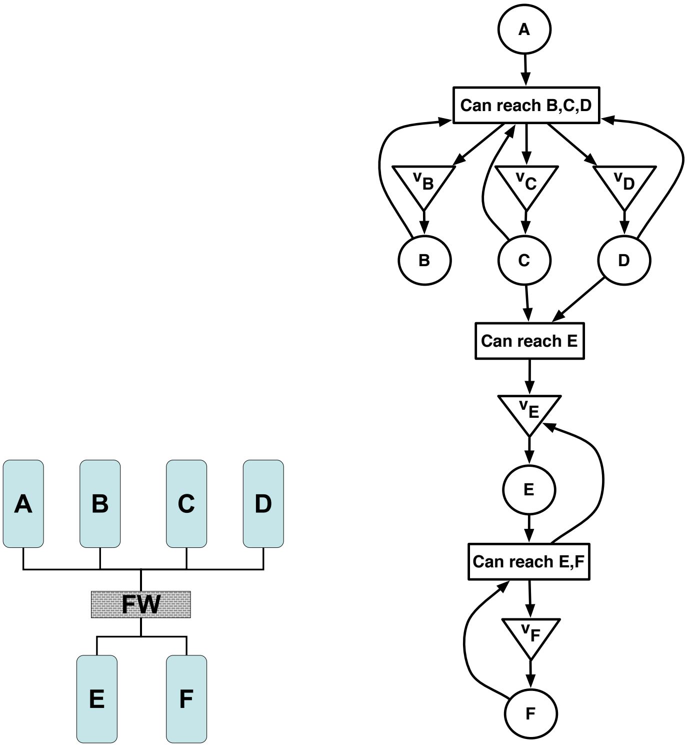
We compute the marginal distribution in (11) using a Monte-Carlo simulation, since computing $P_e(d|a_0, \mathcal{A}, k)$ for the generic problem is NP-complete for reasonably complex networks [21].

The Monte-Carlo algorithm (Appendix A) is a variant of the algorithm for probabilistic networks with random links presented in [21]. In our case, instead of links, vulnerabilities can be randomly exploited or not, but where the exploitation of a vulnerability might require that some other vulnerabilities must already have been exploited. Our algorithm starts from the initial attack state, and then randomly draws an exploitation skill k. Thereafter, we follow the MP attack graph by randomly exploiting all vulnerability instances reachable from the prerequisites of the initial attack state. Next, we repeat the last step for all reachable attack states of the exploited vulnerability instances, until we have tried to exploit all vulnerability instances gained access to through prerequisites. Then, from a large number of repetition of the previous steps, while keep starting the attacks from the initial attack state, we can use our predicates to check for loss of security for any data set instance, and finally, estimate probabilities of successful attacks.

6 Modeling and Analyzing the Simple Network Example

In Table 1-2, we model the simple network example from Sect. 1. The network in Fig. 1(a) is modeled in Table 1 and the MP attack graph shown in Fig. 1(b) is modeled in Table 2. Table 3 and Table 4 shows a data model as well as an impact model respectively. The data model models that data set d_E is located at host E, d_F is located at F and that $d_{E'}$ is dependent of d_E for its existence and that $d_{E'}$ flows from E to C. We let all vulnerabilities have the same required exploitation skill-level.

In Fig. 6, each curve shows the expected risk during the next time slot ($s = 1$) for one of the required exploitation skill-levels. The expected risk is shown on the y-axis (maximum risk is 10.5) and the time on the x-axis. At time point zero, there are no attacks, but because of our prior distributions, we can infer a risk anyway. Thereafter, an attack attempt is recorded at each time point as



(a) The simple example network.

(b) The multi-prerequisites attack graph.

Fig. 1. The simple example from [13]

shown in Table 5. A time slot is 15 time points and thus, the number of time slots $n(T)$ is $1, 2, \ldots, 7$ depending on at what time the risk is computed.

Our prior distribution $p(k) \propto 1$ implies the cautious, prior belief that the exploitation skill is uniformly distributed in the attacker community, while for the prior distribution of λ we set $a = 1$ and $b = 1$ in (11) such that $p(\lambda) = e^{-\lambda}$.

As we might have expected, each curve looks different given a different required exploitation skill-level. Not surprisingly, for all curves, the risk decreases given evidence of unsuccessful attacks, while the risk increases when given evidence of successful attacks. Reasonable enough, the Low-curve is most sensitive to all types of attacks, while attacks at a lower skill-level have lesser impact on a risk curve with a higher required skill-level.

We can now compare our model with the recommendation algorithm for patches in [13]. Table 6 shows the impact estimation when the probability of success is 1.0 for all vulnerabilities. We have compared three different data models and five different patch choices. In [13], patches of set of vulnerabilities are compared. For this simple example, we only patch a single vulnerability instance. Column 2 shows the impact estimation of the models in Table 3 and 4. Column 3

Table 1. Network model

$$\begin{array}{l}
N = \{A, B, C, D, FW, E, F\} \\
E = \{l_1, l_2\} \\
i(FW) = \{i_{FW1}, i_{FW2}\} \\
\forall n \in N_0 : i(n) = \{i_n\} \\
i(l_1) = \{i_A, i_B, i_C, i_D, i_{FW1}\} \\
i(l_2) = \{i_{FW2}, i_E, i_F\} \\
r(FW) = \{\langle i_{FW1}, i_{FW2}, [C, i_C, 0], \\
[E, i_E, 0]\rangle, \langle i_{FW1}, i_{FW2}, [D, i_D, 0], \\
[E, i_E, 0]\rangle\} \\
r(C) = \{\langle i_C, i_C, [C, i_C, 0], [E, i_E, 0]\rangle\} \\
r(D) = \{\langle i_D, i_D, [D, i_D, 0], [E, i_E, 0]\rangle\} \\
r(E) = \{\langle i_E, i_E, [C, i_C, 0], [E, i_E, 0]\rangle, \\
\langle i_E, i_E, [D, i_D, 0], [E, i_E, 0]\rangle\}
\end{array}$$

Table 2. MP attack graph

$$\begin{array}{l}
A = \{a_A, a_B, a_C, a_D, a_E, a_F\} \\
Q = \{q_{BCD}, q_E, q_{EF}\} \\
v(q_{BCD}) = \{v_B, v_C, v_D\} \\
v(q_E) = \{v_E\} \\
v(q_{EF}) = \{v_E, v_F\} \\
q(a_A) = q(a_B) = \{q_{BCD}\} \\
q(a_C) = \{q_{BCD}, q_E\} \\
q(a_D) = \{q_{BCD}, q_E\} \\
q(a_E) = \{q_{EF}\}, q(a_F) = \{q_{EF}\} \\
a(v_B) = a_B, a(v_C) = a_C \\
a(v_D) = a_D, a(v_E) = a_E, a(v_F) = a_F
\end{array}$$

Table 3. Data model

$$\begin{array}{l}
S = \{s_B, s_C, s_D, s_E, s_F\} \\
\forall h \in N - \{A, FW\} : s(h) = \{s_h\} \\
D = \{d_E, d_{E'}, d_F\} \\
store(s_F) = \{d_F\} \\
store(s_E) = \{d_E\} \\
dependencies(s_E, d_{E'}) = \{d_E\} \\
outputs(s_E, i_E) = \{\langle d_{E'}, E, i_E, 0\rangle\} \\
inputs(s_C, i_C) = \{\langle d_{E'}, E, i_E, 0\rangle\}
\end{array}$$

Table 4. Impact Model

$$\begin{array}{l}
V = \{v_B, v_C, v_D, v_E, v_F\} \\
\forall h \in N - \{A, FW\} : v(s_h) = \{v_h\} \\
\forall h \in N - \{A, FW\}, e \in \{c, i, a\} : \\
\quad impact(v_h, e) = Complete \\
\forall e \in \{c, i, a\} : loss(d_E, e) = 1 \\
\forall e \in \{c, i, a\} : loss(d_{E'}, e) = 0.5 \\
\forall e \in \{c, i, a\} : loss(d_F, e) = 2
\end{array}$$

shows the values when we remove the data flow, in which case $d_{E'}$ cannot be reached anymore because v_E is patched. Column 4 has the values for the case when each host $B - F$ has a data set instance with loss value 1. Notice that patching v_E is the best choice in all cases.

The column 3 and 4 of Table 6 can easily be replicated by the algorithm in [13], but not column 2. Column 3 is replicated by letting E have weight 4.5 and F weight 6 and column 4 by letting $B - F$ have weight 1. However, consider the second column, if we use the same weights as for the third column, we would have the result in the first column in Table 7, where the value of patching v_E differs from that in column 2 in Table 6. To fix this, we can assign weight 3 to E, 1.5 to C and 6 to F, but then we get the result in column 2 of Table 7, where instead the value of patching v_C differs from that in column 2 in Table 6. Consequently, the importance of a host depends on the attack graph in case of the original algorithm, while this is not the case for our improved model. The problem is that in order to compromise data set instance $d_{E'}$, we only have to compromise either C or E. Thus compromising one of them is enough. However, this is not possible to express using the simple weight assignment.

In Fig. 6, we show the estimated risk using our statistical model, corresponding to column 1 in Table 6. We let all vulnerabilities have required skill-level *Low* while using the attacks from Table 5. As can be seen, the risk curves of our

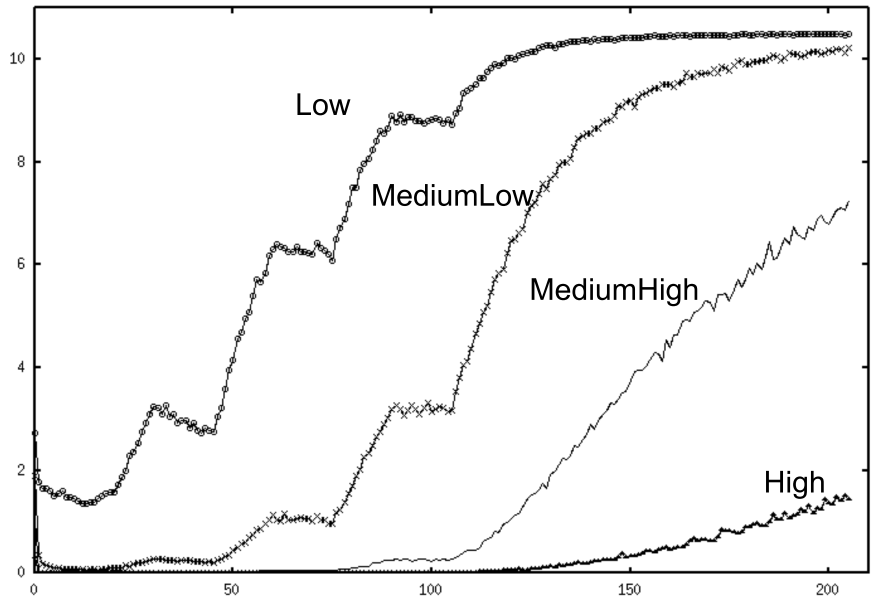


Fig. 2. Risk as function of time according to required exploitation skill-level

Table 5. Attack data; a time slot is 15 time points

Time	Skill-level	Success.	#attacks
0	none	-	0
1 - 15	Low	No	15
16 - 30	Low	Yes	30
31 - 45	MedLow	No	45
46 - 60	MedLow	Yes	60
61 - 75	MedHigh	No	75
76 - 90	MedHigh	Yes	90
90 - 105	High	No	105
106 - 205	High	Yes	205

Table 6. Impact for three different data models

Patch	Unmod	No Flow	Uniform
none	10.5	10.5	5
v_B	10.5	10.5	4
v_C	10.5	10.5	4
v_D	10.5	10.5	4
v_E	1.5	0	3
v_F	4.5	4.5	4

Table 7. Impact for original model

Original 1	Original 2
10.5	10.5
10.5	10.5
10.5	9
10.5	10.5
0	1.5
4.5	4.5

algorithm converge into the same values, given enough attack evidence, as in column 1 in Table 6. Thus similarly, by removing vulnerability instance v_E we would have the least risk regardless of attack data. Thus, v_e would be our best recommendation. The next best would be v_F. However, if we would have made the recommendation before time point 45, we would instead have recommended vulnerability instance v_C as the next best recommendation. Notice also that at time step 100, removing v_D or v_C would be better than removing v_B or doing nothing. Accordingly, in contrast to the original algorithm that produces no difference in risk/impact over time, our algorithm makes it possible to make fine grained prioritization of patches, using not only expert knowledge as input, but also monitor data.

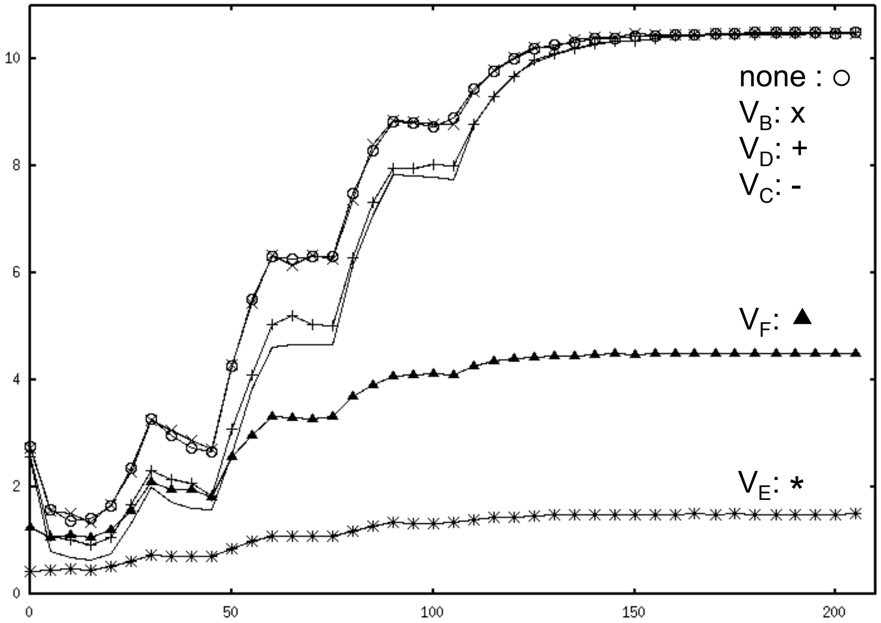


Fig. 3. The risk curves for patching one vulnerability

7 Related work

A set of papers [3,4,5,6,7] define metrics to measure the security of networks based primarily on different ways of weighing paths in an attack graph. Other approaches – like our work – use the risk as a measure of the security of a network where the cost or loss of an intrusion is used as the basis for the metric [8,9,10,11].

An early work in this area is presented in [3,4]. As security metrics the authors use the *time* from an attack starts until it succeeds and the *effort* required by the attacker to succeed. They use a Markov model over a privilege graph to compute the two metrics, where they assume exponential distributions for the required time and effort. Their metrics are complementary to our work in that they consider the *cost* for an attacker, while we consider the loss to the system owners.

A work inspired by reliability analysis is presented in [5,6]. The authors use a model checker to construct an attack graph from a detailed model of the system. As a security metric they propose the probability that an attacker will end up in a unsafe system state. They estimate the probabilities by the use of a Markov Decision Process (MDP) [22]. In contrast to our work, they don't consider the different skill required to use an exploit. In addition, they do not suggest how to come up with estimations of the transition probabilities of the MPD.

In [7] the authors use the PageRank algorithm of Google to measure the security of a network from an attack graph. The result is an ergodic Markov chain that converges into a probability distribution over the attack states. In contrast to our work, they do not consider the value of protected assets or dependencies between attack states; neither do they suggest how to use historical data to derive transition probabilities.

In [8], the authors present a framework called RheoStat that uses a risk-based analysis to select a response. Due to that RheoStat's likelihood metric is not probability based, RheoStat can only estimate the risk conditioned on the current intrusion alerts, while our work also can estimate the risk given the intrusion activities over time. Thus, our work can be used to compare different network configurations.

The work in [9] uses a Hidden Markov Model (HMM) to estimate the probability that nodes in a network are in malicious states given observed intrusion alerts from an IDS. In contrast to our work, the authors only assign costs of each host being in a malicious state. The impact is estimated either as the total expected cost for all hosts or the average expected cost per host. They neither consider the value of protected assets nor dependencies between host.

Another approach to real-time risk analysis uses Hidden Markov Models as input to a Fuzzy inference system [10]. A set of Fuzzy rules are used to derive three linguistic variables: intrusion frequency, probability of threat success and severity. The HMMs provide an estimation of the intrusion frequency of different attack types, while the other input values come from a distributed intrusion detection system and from a traffic rate monitor. Then another set of Fuzzy rules infer the final risk assessment from the three linguistic variables. In [11], the same authors present an extension where the Fuzzy rules are optimized using a neural network that learn from given training examples. The papers are brief on how the system works and how it was tested, hence a comparison with our approach is not easy to do.

8 Summary and Concluding Remarks

We have presented a framework for assessing the security risk to a network using data flows over an attack graph and a Bayesian model of the exploitation skill of the attacker community, given a record of attack data. By modeling the flow of data, we take into account dependencies between different hosts and different data flows. In addition, we are able to update the risk computation as soon as we receive more attack data.

We also propose using a honeypot to collect historical attack data. However, in order to model the exploitation skill, each vulnerability instance must be classified by an expert into a required exploitation skill-level. This can turn into a problem since a vulnerability that once was hard to exploit might suddenly become easy when somebody creates a downloadable exploitation code. Thus, some monitoring system for updating the information about vulnerabilities is needed as well as a support system for classifying old and new vulnerabilities given new information.

Acknowledgements. This work has been performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab Systems, TeliaSonera and T2Data.

References

1. Nessus: The network vulnerability scanner (April 2009), `http://www.nessus.org`
2. Lippman, R., Ingols, K.: An annotated review of past papers on attack graphs. Project Report PR-IA-1, MIT Lincoln laboratory (March 2005)
3. Dacier, M., Deswarte, Y., Kaaniche, M.: Quantitative assessment of operational security: Models and tools (1996)
4. Ortalo, R., Deswarte, Y.: Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Transactions on Software Engineering 25, 633–650 (1999)
5. Jha, S., Sheyner, O., Wing, J.M.: Minimization and reliability analyses of attack graphs. Technical Report CMU-CS-02-109, School of Computer Science, Carnegie Mellon University (2002)
6. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: IEEE Symposium on Security and Privacy, pp. 273–284 (2002)
7. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 127–144. Springer, Heidelberg (2006)
8. Gehani, A., Kedem, G.: RheoStat: Real-time risk management. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 296–314. Springer, Heidelberg (2004)
9. Årnes, A., Valeur, F., Vigna, G., Kemmerer, R.A.: Using hidden markov models to evaluate the risks of intrusions - system architecture and model validation. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 145–164. Springer, Heidelberg (2006)
10. Haslum, K., Abraham, A., Knapskog, S.: Dips: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. In: Third International Symposium on Information Assurance and Security. IEEE Computer Society Press, Los Alamitos (2007)
11. Haslum, K., Abraham, A., Knapskog, S.J.: Hinfra: Hierarchical neuro-fuzzy learning for online risk assessment. In: Asia International Conference on Modelling and Simulation, pp. 631–636 (2008)

12. Stoneburner, G., Goguen, A., Feringa, A.: Risk management guide for information technology systems. Technical Report NIST SP 800-30, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology (2002)
13. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Computer Security Applications Conference, pp. 121–130 (2006)
14. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis, 2nd edn. Chapman & Hall/CRC, Boca Raton (July 2003)
15. CVSS: Common vulnerability scoring system (April 2009), http://www.first.org/cvss/
16. Olsson, T.: Impact estimation using data flows over attack graphs. In: Proceedings of the 14th Nordic Conference on Secure IT Systems, NordSec (2009)
17. NVD: National vulnerability database (April 2009), http://nvd.nist.gov/
18. Abramowitz, M., Stegun, I.A. (eds.): Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Dover, New York (1972)
19. McGrew, R., Vaughn, R.B.: Experiences with honeypot systems: Development, deployment, and analysis. In: Proceedings of the 39th Hawaii International Conference on System Sciences (2006)
20. Alata, E., Nicomette, V., Kaâniche, M., Dacier, M., Herrb, M.: Lessons learned from the deployment of a high-interaction honeypot. In: EDCC 2006, 6th European Dependable Computing Conference (October 2006)
21. Marzot, G.S.: Netstat: A probabilistic network connectivity analysis tool. Technical Report A838262, MITRE CORP BEDFORD MA (1993)
22. Filar, J.A., Vrieze, K.: Competitive Markov Decision Processes. Springer, Heidelberg (1996)

A Monte-Carlo Algorithm

```
Input: $a_0$ - attack state, $d$ - data set instance, $s$ - time slots, $e$ - security aspect,
       $X$ - attack data
Output: $prob$ - the estimated probability of loss of security.
$N \leftarrow 100000$ : Number of sampled values.
$m_e \leftarrow 0$ : mean value of the probability of loss of security .
$x \leftarrow 0$ : a weighted value, indicating whether $d$ was compromised.
$m_{P(X)} \leftarrow 0$ : estimated mean value for probability of the $Data$.
$s_{P(X)} \leftarrow 0$ : estimated standard deviation of probability of the $Data$.
for $i = 0; i < N; i = i + 1$ do
    $k \leftarrow$ sample from $p(k)$;
    $StatesCurrent \leftarrow \{a_0\}$, $VulnsCurrent \leftarrow \emptyset$, $StatesUsed \leftarrow \emptyset$;
    $VulnsUsed \leftarrow \emptyset$, $PrerequistesUsed \leftarrow \emptyset$, $VulnsExploited \leftarrow \emptyset$;
    Collect samples of successfully exploitations:
    while $StatesCurrent \neq \emptyset$ do
        foreach $a \in StatesCurrent$ do
            foreach $q \in q(a)$ and $q \notin PrerequistesUsed$ do
                foreach $v \in v(q)$ and $v \notin VulnsUsed$ do
                    $VulnsCurrent \leftarrow VulnsCurrent \cup \{v\}$;
                $VulnsUsed \leftarrow VulnsUsed \cup v(q)$;
            $PrerequistesUsed \leftarrow PrerequistesUsed \cup q(a)$;
        $StatesUsed \leftarrow StatesUsed \cup StatesCurrent$, $StatesCurrent \leftarrow \emptyset$;
        foreach $v \in VulnsCurrent$ do
            if $rand(0,1) > 1 - \Phi(k, m_{z(v)}, \sigma_{z(v)})$ and $a(v) \notin StatesUsed$ then
                $StatesCurrent \leftarrow StatesCurrent \cup \{a(v)\}$,
                $VulnsExploited \leftarrow VulnsExploited \cup \{v\}$;
        $VulnsCurrent \leftarrow \emptyset$;
    Estimate current mean value:
    if $lossOfSecurity(e, d, VulnsExploited)$ holds true then
        $x \leftarrow \left(\frac{n(T)+b}{n(T)+s+b}\right)^{n(X)+a} \cdot p(X|k)$ else $x \leftarrow P(X|k)$;
    if $i = 0$ then
        $m'_e \leftarrow x$, $s'_e \leftarrow 0$, $m'_{P(X)} \leftarrow P(X|k)$, $s'_{P(X)}) \leftarrow 0$ : Init old values
    else
        $m_e \leftarrow m'_e + (x - m'_e)/(i+2)$,
        $m_{P(X)} \leftarrow m'_{P(X)} + (P(X|k) - m'_{P(X)})/(i+2)$
        $m'_e \leftarrow m_e$, $s'_e \leftarrow s_e$, $m'_{P(X)} \leftarrow m_{P(X)}$, $s'_{P(X)} \leftarrow s_{P(X)}$ : Save old
        values
$prob \leftarrow 1 - \frac{m_e}{m_{P(X)}}$;
```