Ranking Attack Graphs*

Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu,
Edmund Clarke, and Jeannette Wing

Carnegie Mellon University, Pittsburgh, USA
{vaibhav, cbartzis, haifengz, emc, wing}@cs.cmu.edu

Abstract. A majority of attacks on computer systems result from a combination of vulnerabilities exploited by an intruder to break into the system. An Attack Graph is a general formalism used to model security vulnerabilities of a system and all possible sequences of exploits which an intruder can use to achieve a specific goal. Attack Graphs can be constructed automatically using off-the-shelf model-checking tools. However, for real systems, the size and complexity of Attack Graphs greatly exceeds human ability to visualize, understand and analyze. Therefore, it is useful to identify relevant portions of an Attack Graph. To achieve this, we propose a ranking scheme for the states of an Attack Graph. Rank of a state shows its importance based on factors like the probability of an intruder reaching that state. Given a Ranked Attack Graph, the system administrator can concentrate on relevant subgraphs to figure out how to start deploying security measures. We also define a metric of security of the system based on ranks which the system administrator can use to compare Attack Graphs and determine the effectiveness of various defense measures. We present two algorithms to rank states of an Attack Graph based on the probability of an attacker reaching those states. The first algorithm is similar to the PageRank algorithm used by Google to measure importance of web pages on the World Wide Web. It is flexible enough to model a variety of situations, efficiently computable for large sized graphs and offers the possibility of approximations using graph partitioning. The second algorithm ranks individual states based on the reachability probability of an attacker in a random simulation. Finally, we give examples of an application of ranking techniques to multi-stage cyber attacks.

Keywords: Google PageRank, Attack Model, Attack Graph, Model Checking, security metric.

* This research was sponsored by the Office of Naval Research under grant no. N00014-01-1-0796, the Army Research Office under grant no. DAAD19-01-1-0485, and the National Science Foundation under grant nos. CNS-0411152, CCF-0429120, and 0433540. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

D. Zamboni and C. Kruegel (Eds.): RAID 2006, LNCS 4219, pp. 127–144, 2006.

1 Introduction

A large computer system builds upon multiple platforms, runs diverse software packages and supports several modes of connectivity. Despite the best efforts of software architects and coders, such systems inevitably contain a number of residual faults and security vulnerabilities. Hence, it is not feasible for a system administrator to try and remove each and every vulnerability present in these systems. Therefore, the recent focus in security of such systems is on analyzing the system globally, finding attacks which are more likely and severe, and directing resources efficiently to increase confidence in the system.

To evaluate security of such a system, a security analyst needs to take into account the effects of interactions of local vulnerabilities and find global vulnerabilities introduced by interactions. This requires an appropriate modeling of the system. Important information such as the connectivity of elements in the system and security related attributes of each element need to be modeled so that analysis can be performed. Analysis of security vulnerabilities, the most likely attack path, probability of attack at various elements in the system, an overall security metric etc. is useful in improving the overall security and robustness of the system. Various aspects which need to be considered while deciding on an appropriate model for representation and analysis are: ease of modeling, scalability of computation, and utility of the performed analysis.

There has been much work on modeling specific systems for vulnerability analysis. Zhu [24] models computer virus infections using an Infection Graph, where nodes represent hosts and an arc represents the probability of transfer of a virus from source to target host independent of the rest of the system. Infection Graphs are used to find the most vulnerable path for virus infection on a particular host. Ortalo *et al.* [19] describe a methodology for modeling known Unix security vulnerabilities as a Privilege Graph, where a node represents the set of privileges owned by a user and an arc represents grant of an access privilege. Dawkins and Hale [5] present a multi-stage Network Attack Model which contains a DAG (Directed Acyclic Graph) similar to the Infection Graph in [24]. However, their model is generalized for different kinds of attacks and an XML description is proposed. Sheyner *et al.* [21,9,22] present a data structure called an Attack Graph to model the security vulnerabilities of a system and their exploitation by the attacker. An Attack Graph is a succinct representation of all paths through a system that end in a state where an intruder has successfully achieved his/her goal. An attack is viewed as a violation of a safety property of the system, and off-the-shelf model checking [3] techniques are used to produce Attack Graphs automatically.

Various techniques for quantitative security analysis are presented in [4,10,20,17]. Dacier *et al.* [4] use Privilege Graphs to model the system, therefore restricting the analysis to a specific family of attacks. Empirical and statistical information is used to estimate the time and effort required for each type of attack. MTTF (mean time to failure) is computed as a metric of the security level of a system. The framework proposed in [20] requires attacker profiles and attack templates with associated probabilities as part of the input. An ad hoc algorithm is used to generate Attack Graphs. Using a modified shortest path algorithm, the most likely attack sequences are computed. Madan *et al.* [17] give a theoretical description of various methods which can be used to quantify security based attributes of an intrusion tolerant system. Security intrusion and response of an intrusion tolerant system are modeled using a Semi-Markov Process (SMP). Security quantification analysis is carried out to compute measures like steady state availability, mean time to failure and probabilities of security failure due to violations of different security attributes. However, the analysis is based on the availability of values for various model parameters and is feasible for small Markov Chains only. Another related approach is the one described in [8], where Alternating Probabilistic Attack Graphs are used for analysis. However, the system designer has to provide a priori probabilities for most events in the system.

We propose a ranking scheme for the states of an Attack Graph. Rank of a state shows its importance based on factors such as the probability of an intruder reaching the state. The framework we propose is summarized in Figure 1. First we obtain a formal description of the system to be analyzed, an Attack Model, that captures all possible behaviors of the system as it interacts with possibly malicious peers. Given a security property, we then model check the Attack Model, thus obtaining a compact description of all executions that violate the security property, an Attack Graph. At the same time, we apply a ranking algorithm to the state transition graph of the Attack Model to compute the ranks of its states. We present two ranking algorithms to rank states based on the probability of an intruder reaching those states. The first algorithm is similar to the PageRank algorithm [2,1,15] used by Google. The second algorithm ranks states based on the reachability probability of an intruder in a random simulation. As there is a direct correspondence between the states of an Attack Model and an Attack Graph, we also get the ranks of states of the Attack Graph. The Ranked Attack Graphs are valuable for a system administrator as they allow him to estimate the security level of the system and provide a guide for choosing appropriate corrective or preventive measures.

The main advantages of our approach are:

- **Ease and flexibility of modeling** : Finding ranks using our technique does not necessarily require a priori probabilities for all events. If the probabilities are available, then we can use them for more accurate modeling. Even if the exact probabilities are not available, modeling the attacks randomly is expected to perform as good as PageRank performs on the World Wide Web graph. In realistic situations, an attacker very rarely has complete information about the network and the attack mostly proceeds using a repeated scan-probe approach. This is similar to a websurfer navigating across webpages on the World Wide Web by following hyperlinks. Likewise, in the case of automated attacks by computer viruses and worms, the attacks are random in nature [25,23].

  The ranking technique we use is also very flexible, since we can model different knowledge levels of the attacker and his intentions by simply adding

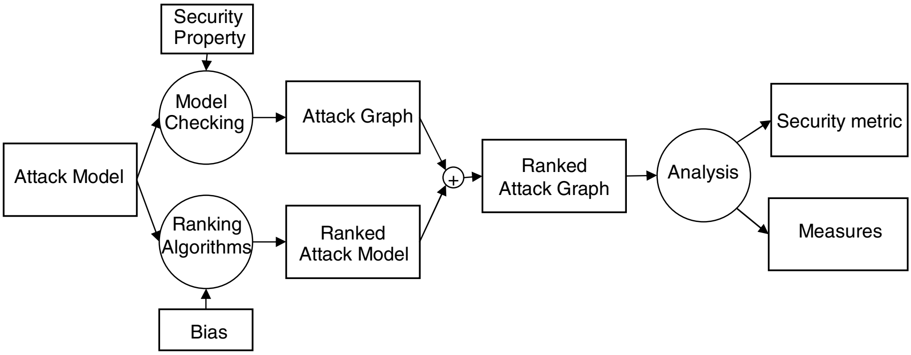


Fig. 1. Ranking to analyze security of Attack Models

a bias in the rank computation. This is similar to personalization [7,1,15] of PageRank. Moreover, we can combine the ranks obtained using this algorithm with other criteria for ranking states such as the severity of different failures in the system.

- **Scalability** : There exist efficient algorithms to compute PageRank over graphs containing billions of webpages. Techniques based on sparse matrices [6], extrapolation methods [13], adaptive methods [11], hierarchical block rank computation [12], aggregation methods [16] are used to accelerate the computation of PageRank. Since one of our algorithms is similar to PageRank, we can also handle state transition graphs of comparable sizes.
- **Applicability to a variety of situations** : Attack Graphs and Attack models are a very general formalism which can be used to model a variety of situations and attacks. The system under attack could be anything: a computer network under attack by hackers, a city under siege during war, an electric grid targeted by terrorists. Moreover, Attack Graphs can be automatically generated using off-the-shelf model checking techniques. Hence, a variety of situations can potentially be modeled and analyzed.
- **Useful Analysis** : A number of useful analyses can be carried out over the Ranked Attack Graphs, which help a system administrator determine the security level of the system and decide amongst various possible defense measures. Ranks provide a detailed security metric which can be subsequently used by the system architect / administrator. Ranks of states can be used to determine the probability and severity of security failures at various elements in the system. For realistic examples, the size and complexity of Attack Graphs greatly exceeds the human ability to understand and analyze. Ranks provide a way to determine the relevant parts of the Attack Graph to figure out how to best deploy security measures.

In the next section, we give formal definitions of an Attack Model and an Attack Graph. In Section 3, we explain the algorithms used to rank states of an

Attack Graph. In Section 4, we describe various analyses that can be performed on the Ranked Attack Graphs. In Section 5, we give examples of applying our ranking techniques to real-life systems and applications.

2 Attack Models and Attack Graphs

Sheyner *et al.* [21] introduced the concept of Attack Models and Attack Graphs to model the security vulnerabilities of a system and their exploitation by an attacker.

An **Attack Model** is a formal representation of security related attributes of the attacker, the defender and the underlying system. Formally,

Definition 1. *Let AP be a set of atomic propositions. An* Attack Model *is a finite automaton* $M = (S, \tau, s_0, l)$*, where S is a set of states,* $\tau \subseteq S \times S$ *is the transition relation,* $s_0 \in S$ *is the initial state, and* $l : S \rightarrow 2^{AP}$ *is a labeling of states with the set of propositions true in that state.*

A state in the model is a valuation of variables describing the attacker, the defender and the system. The transitions in the system correspond to actions taken by an attacker which lead to a change in the overall state of the system. The starting state of the model denotes the state of the system where no damage has occurred and the attacker has just entered the system using an entry point. As an example, if we consider the case of a computer network Attack Model, a state represents the state of the intruder, the system administrator and the network of computers. The transitions correspond to actions of the attacker such as running a network scan, probing a computer for vulnerabilities and exploiting vulnerabilities to get more privileges on that computer.

An **Attack Graph** is a subgraph of an Attack Model, which consists of all the paths in an Attack Model where the attacker finally succeeds in achieving his goal. Formally,

Definition 2. *Let AP be a set of atomic propositions. An* Attack Graph *or AG is a finite automaton* $G = (S, \tau, s_0, E, l)$*, where S is a set of states,* $\tau \subseteq S \times S$ *is the transition relation,* $s_0 \in S$ *is the initial state,* $E \subseteq S$ *is the set of error states, and* $l : S \rightarrow 2^{AP}$ *is a labeling of states with the set of propositions true in that state.*

Given an Attack Model, model checking techniques are used to generate Attack Graphs automatically. The negation of the attacker's goal is used as the correctness property during model checking. These properties are called *security properties*. An example of a security property in computer networks would be "the intruder cannot get root access on the main web server". A model checker is used to find out all states in the Attack Model where the security property is not satisfied. We call these states *error states*, comprising set E. An Attack Graph is a subgraph of the Attack Model which only contains paths leading to one of the error states. [21,9,22] describe the details of the algorithm to construct an Attack Graph, given an Attack Model and a security property.

In order to be able to find the reachability probability of various states in an Attack Model, we associate probabilities with transitions in the model. We call the resulting model a **Probabilistic Attack Model**. Formally,

Definition 3. *A* Probabilistic Attack Model *is a 4-tuple* $M = (S, \tau, s_0, l)$*, where* S *is a set of states,* $\tau : S \times S \rightarrow [0, 1]$ *is the transition relation such that* $\forall s \in S, \sum_{s' \in S} \tau(s, s') = 1$*,* $s_0 \in S$ *is the initial state, and* $l : S \rightarrow 2^{AP}$ *is a labeling of states with the set of propositions true in that state.*

3 Two Ranking Algorithms

We first describe the theory of the basic PageRank algorithm used to rank webpages on the World Wide Web. Then, we give a slightly modified version of the PageRank algorithm to rank states of an Attack Graph. We also provide an alternative algorithm for ranking states of an Attack Graph based on the reachability probability of an attacker in a random simulation.

3.1 Using PageRank to Rank Attack Graphs

Google's PageRank Algorithm. PageRank is the algorithm used by Google to determine the relative importance of webpages on the World Wide Web. PageRank is based on a model of user behaviour. It assumes there is a "random surfer" who starts at a random webpage and keeps clicking on links, never hitting the 'back' button, but eventually gets bored and starts on another random page. The computed rank of a page is the probability of the random surfer reaching that page. PageRank can be interpreted as a *Markov Process*, where the states are pages, and the transitions are the links between pages which are all equally probable. To capture the notion that a random surfer might get bored and *restart* from another random page, a *damping factor* d is introduced, where $0 < d < 1$. The transition probability from a state is divided into two parts: d and $1 - d$. The d mass is divided equally among the state's successors. Random transitions are added from that state to all states with the residual probability of $1 - d$ equally divided amongst them. If the random surfer arrives at a page with no hyperlinks (called a*dangling state*), he picks another page at random and *restarts* from there. So, new hyperlinks are added from a dangling state to all other states with the transition probability equally divided amongst them. In what follows, these links are treated as ordinary links.

Consider a web graph with N pages linked to each other by hyperlinks. Let $Out(j)$ be the set of outlinks (hyperlinks) from page j and $In(j)$ be the set of pages linking to page j. After a sufficiently long period of time, the probability π_i of the surfer being at page i is given by:

$$\pi_i = \frac{1-d}{N} + d \sum_{j \in In(i)} \frac{\pi_j}{|Out(j)|} \qquad (1)$$

The first term in the equation corresponds to the probability transferred to a state from the random transitions. The second term represents the probability transferred to a state from its predecessors and dangling states. Let $\mathbf{R} = (r_1, r_2, .., r_N)^T$ be the PageRank vector, where r_i is the rank of page i. The PageRank of page i is defined to be the probability π_i as used in Eq. 1. Because Eq. 1 is recursive, it must be iteratively computed until π_i converges. The PageRank values are the entries of the dominant eigenvector of the modified and normalized adjacency matrix $\mathbf{Z}$:

$$\mathbf{Z} = (1-d)\left[\tfrac{1}{N}\right]_{N \times N} + d\mathbf{A}, \tag{2}$$

$$\text{where} \quad A_{ij} = \begin{cases} \frac{1}{|Out(j)|} & \text{if there is an edge from j to i} \\ 0 & \text{otherwise} \end{cases}$$

One iteration of Eq. 1 is equivalent to computing $\mathbf{R^{t+1}} = \mathbf{ZR^t}$. After convergence, we have $\mathbf{R^{F+1}} = \mathbf{R^F}$, or $\mathbf{R^F} = \mathbf{ZR^F}$, which means $\mathbf{R^F}$ is an eigenvector of $\mathbf{Z}$.

Various issues like the complexity of computation, storage, stability, and convergence of PageRank have been extensively studied [1,15] . A number of techniques based on sparse matrix computations, matrix permutations, utilization of the block structure of the WWW etc. have been developed, allowing efficient computation of PageRank over large web graphs [6,7,11,12,13,16]. PageRank for a 100 million webpages can be computed in a few hours on a medium size workstation.

Ranking States of an Attack Graph. Given an Attack Model $M = (S, \tau, s_0, L)$, we first construct a Probabilistic Attack Model for the system. As described in Section 2, a transition from a state to another represents an atomic attack or event that leads to a change in the overall system state. We assign probabilities to these events so that the probability for the whole system to converge to a certain state after a long run can be computed. Similar to the Google PageRank algorithm, we divide each state's probability into d and $1-d$, where d is a parameter to be tuned (termed as damping factor in PageRank). We further divide the d mass equally among the state's successors. This is a realistic assumption, since many known attacks use a brute force probe-scan approach. Automated attacks, such as viruses and worms [25,23], are often designed to behave randomly. In addition, we add a transition from each state pointing back to the initial state with probability $1-d$, modeling that at any time there is a chance that either the attacker will abort the current attack and try a different way to attack the system or that the attack may be discovered and the system administrator will isolate the system and recover it. This is different from the basic PageRank algorithm where for each state, edges are added towards all other states to model that at any time the web surfer may randomly select any other webpage in the world from which to continue surfing. In the case of Attack Models, it is unreasonable to assume that the attacker can move the system to any arbitrary state by jumping into the middle of an attack.

Now the Probabilistic Attack Model is established, and we want to find out the long term probability π_j for the system to arrive at a certain state j from the initial state. We define the rank of a state in the Probabilistic Attack Model to be equal to the probability π_j. With probability theory, π_j may not always exist, or may not be interesting (for example, maybe 0). A detailed proof is available in the appendix to justify that for the Probabilistic Attack Model constructed with the above mentioned method, π_j always exists. We prove that the Probabilistic Attack Model corresponds to an ergodic Markov Chain, which is a necessary condition for the iterative computation to converge. We compute the ranks of all states using the method for computing PageRank described in Section 3.1.

Now, we label the states of the Attack Graph with the ranks obtained above. The states in an Attack Graph are a subset of the states in the Probabilistic Attack Model. Hence, every state in the Attack Graph is labeled with the rank of the corresponding state in the Probabilistic Attack Model. We are particularly interested in the rank of error states in the Attack Graph. High probability (or rank) of error states means that the system is insecure. Section 4 describes different kinds of analyses possible using a Ranked Attack Graph.

3.2 Alternative Algorithm for Ranking Attack Graphs

Kuehlmann *et al.* [14] give a method to rank states in a state transition graph to guide state space search. The rank of a particular state gives the probability of reaching one of the target states starting from that state in a random simulation run. The ranks are computed using a random walk based strategy. Our alternative ranking algorithm to rank states of an Attack Graph is a modification of the algorithm given in [14]. The rank of a particular state here gives the probability of reaching that state starting from the initial state in a random simulation run.

We give a brief description of the algorithm. Given an Attack Model $M = \{S, \tau, s_0, l\}$ with k states, we construct the transition probability matrix P. Let p_{ij} denote the probability of transition from state s_j to s_i. We define the transition probability p_{ij} as the reciprocal of the number of successors of the state s_j. If the exact transition probability is known, we use that instead. Let $s = (s_1, ...s_k)^T$ be a vector where $s_i = 1$ if s_i is a start state, 0 otherwise. The vector $r^{(m)}$ representing the reachability probabilities for all states in a random simulation run of length up to m is given by:

$$r^{(m)} = \sum_{n=0}^{m} P^n s \tag{3}$$

We consider a set of finite simulation runs with a given distribution of their lengths. We assume a geometric distribution of lengths, for which the number of simulation steps m is given by the following probability mass function:

$$d(m) = \frac{1-\eta}{\eta}\eta^m \quad \text{for } m > 0 \text{ and } 0 < \eta < 1 \text{ constant} \tag{4}$$

We use a geometric distribution of simulation lengths as we believe that longer attacks are less probable. Also, using this distribution, error states closer to the initial state and hence, easier to be attacked, are ranked higher. Let r_i be the reachability probability of state s_i. The following formula computes the reachability probability $r = (r_1, ..., r_k)^T$.

$$r \;=\; \frac{1-\eta}{\eta} \sum_{m=1}^{\infty} \eta^m \sum_{n=0}^{m} P^n s \qquad (5)$$

The reachability probability r_i is defined as the rank of state s_i in the Attack Model. As the states of an Attack Graph are a subset of the states in an Attack Model, we label each state of the Attack Graph with the rank of the corresponding state in the Attack Model. Thus, we obtain a Ranked Attack Graph.

4 Using Ranked Attack Graphs for Security Analysis

1. **Security Metric:** The total rank of all error states provides a good measure of security of the system. By the model checking terminology *error state*, we mean a state whose certain user-specified security property is violated and an undesirable situation happens in the system. If the error states have tiny ranks such as 0.001 or 0.002 for example, it indicates that the whole system is secure enough, thus increasing the confidence of the system users.
2. **Security Improvement:** A system administrator can apply different defense measures with the objective of reducing the total rank of error states. For example, a system administrator may change security policies, or add hardware/software/human at certain components (for example a computer host) of the system, and observe the reduction in the total rank of error states. By these experiments, the administrator is able to improve the security level of the system to a customized desirable level.

   It would be better for a system administrator to eliminate all the highly ranked error states. This can be achieved by making local changes to the Attack Graph which bring about a reduction in the rank of the highly ranked error states. For example, the system administrator can stop a service at a computer or add an intrusion detection component such that it removes some of the incoming transitions of the highly ranked error states in the Attack Graph. This would lead to a reduction in the rank of the highly ranked error states.
3. **Derived Analysis:** In addition, more analysis can be derived from the ranks. One such example is the probability of a host being attacked. By summing up the ranks of states in which a particular host is attacked, the probability of this host being attacked is found. This type of analysis was shown to be useful in some situations such as computer virus attacks [24]. By reducing the infection probability of a certain host through anti-virus measures, it was shown that the epidemic probability can be reduced. Such analysis is also useful to identify the weak grid in a power system targeted by terrorists.

4. **Aid in Visual Analysis:** Attack Graphs suffer from a visual complexity problem. For real situations, the size and complexity of Attack Graphs greatly exceeds the human ability to understand and analyze. Ranks help in viewing more important areas of the Attack Graph selectively. The administrator can adjust the number of states being viewed based on a cutoff on ranks of those states. The administrator could just focus on portions of the Attack Graph containing highly ranked error states and make local changes to get rid of the highly ranked error states.

5 Examples/Applications

In this section, we show applications of ranking techniques to realistic systems. In the example, we consider multi-stage cyber attacks against a network of computers. We construct a Network Attack Graph for a computer network and rank its states to analyze the network for security.

5.1 Ranking Network Attack Graphs

A Network Attack Model is constructed using security related attributes of the attacker and the computer network. Below is a list of components from a network used to construct a network model:

- H, a set of hosts connected to the network. Hosts are computers running services, processing network requests and maintaining data. A host h ϵ H is a tuple (*id, svcs, sw, vuls*) where *id* is a unique host identifier, *svcs* is a list of services active on the host, *sw* is a list of other software running on the host, and *vuls* is a list of host-specific vulnerable components.
- C, a connectivity relation expressing the network topology and inter-host reachability. C is a ternary relation C $\subseteq$ H $\times$ H $\times$ P, where P is a set of integer port numbers. C(h1, h2, p) means that h2 is reachable from h1 on port p.
- TR, a relation expressing trust between hosts. Trust is a binary relation TR $\subseteq$ H $\times$ H. TR(h1, h2) means that a user on h1 can log in on h2 without authentication.
- I, a model of the intruder. We assume the intruder does not have global information about the network such as knowledge of all the possible attacks on the network. The intruder is associated with a function $plvl : H \rightarrow \{none, user, root\}$ which gives the level of privilege of the intruder on each host.
- A, a set of individual actions that the intruder can perform during an attack.

A finite state Attack Model is constructed using the above information about the computer network. A state of the model corresponds to a valuation of variables of each of the above components. The initial state corresponds to the case in which the intruder has *root* privileges on his own machine and no other host. Starting from the initial state, breadth first search is performed to find the set of reachable states and construct the Network Attack Model. In a particular state, we find the set of enabled actions for the intruder. For each action, there is a state transition from the current state to a state which reflects the changes according to the effects of the chosen atomic action. Thus, we obtain a Network Attack Model from the description of the network.

Given a Network Model obtained as above and a security property, model checking is done to obtain a Network Attack Graph. The security property is the negation of the intruder's goal which could be administrative access on a critical host, access to a database, service disruption etc. Network Attack Graphs represent a collection of possible penetration scenarios in a computer network, each culminating in a state where the intruder has successfully achieved his goal.We use the PageRank algorithm of Section 3.1 to produce a Ranked Network Attack Graph.

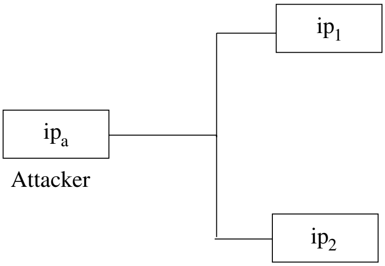
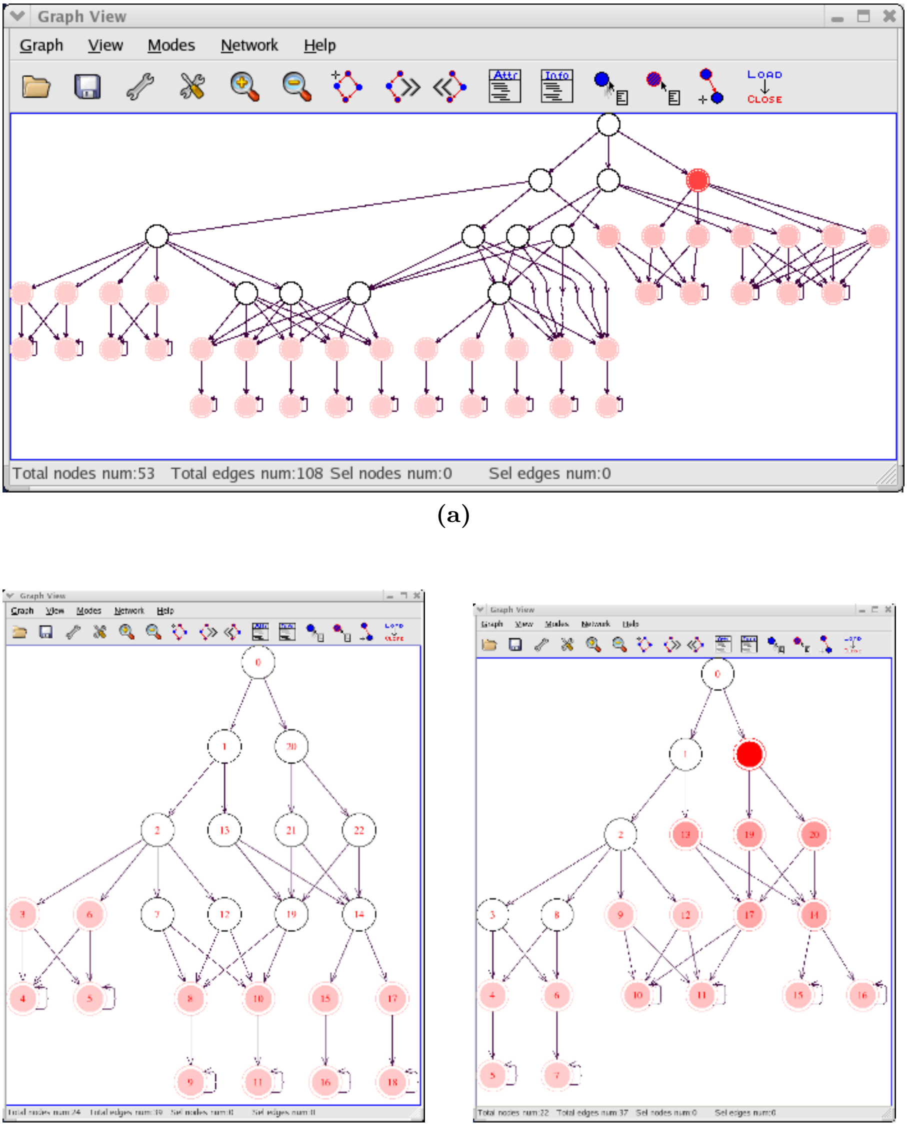


Fig. 2. Computer network A

5.2 Examples of Ranked Network Attack Graphs

We show screenshots of a few examples of Network Attack Graphs. States in the graph have been ranked according to the ranking algorithm based on PageRank. We set the damping factor to 0.85, which is the value Google uses. For each error state, the intensity of color is proportional to the relative rank of that state in the Attack Graph. The security metric based on the total rank of error states is a quantitative guide for comparing Attack Graphs. A system administrator could fix a particular security property, make changes to his network configuration and compare the Attack Graphs obtained using this security metric. Thus, he can determine the relative utility of different security measures. He could also fix the system model and observe changes in the ranks of the Attack Graph based on varying the security property from a weak to a strong one. For example, consider the computer network shown in Figure 2 which has interconnected computer hosts with some services and software vulnerabilities on each host. Let the security property used by the system administrator be " Intruder cannot get root access on ip_2". Figure 3(a) shows the Attack Graph of the network with respect to the above security property. The total rank of error states in the Attack Graph is 0.24. Now, suppose the administrator stops the *sshd* service



Total nodes num:53 Total edges num:108 Sel nodes num:0 Sel edges num:0

(a)

Total nodes num:24 Total edges num:39 Sel nodes num:0 Sel edges num:0

(b)

Total nodes num:22 Total edges num:37 Sel nodes num:0 Sel edges num:0

(c)

Fig. 3. Comparison of Ranked Network Attack Graphs. (a) Attack Graph of the computer network A (b) Attack Graph after stopping service sshd on ip_2 (c) Attack Graph with changed security property.

running on the host ip_2. Figure 3(b) shows the Attack Graph corresponding to the changed network configuration. The total rank of error states in the changed Attack Graph is .053, which shows that the network becomes relatively more secure. Now, suppose the administrator also changes the security property to

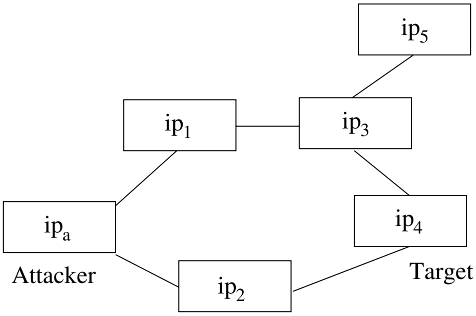


Fig. 4. Computer network B

"Intruder cannot get root access on ip_1". Figure 3(c) shows the Attack Graph of the network with respect to the changed security property. The total rank of error states in the Attack Graph is 0.31. This shows that host ip_1 is more likely to be attacked than host ip_2 in the changed network configuration.

For realistic examples, the size and complexity of Attack Graphs greatly exceeds the human ability to visualize and understand [18]. Ranks provide a solution to this problem by showing the relevant regions of the Attack Graph to be analyzed. Consider the computer network shown in Figure 4. Let the security property used by the system administrator be "Intruder cannot get root access on ip_4". Figure 5 shows the Attack Graph of the network with respect to the above property. The Attack Graph is huge and hence difficult for a human to analyze visually. Our ranking tool highlights the relevant regions of the Attack Graph so that the system administrator can start looking to figure out the best way of deploying security measures. Our visualization tool allows him to zoom-in on portions of the graph with highly ranked states, e.g., the regions depicted in Figure 5. Based on the incoming transitions of the highly ranked error states in these two regions, the system administrator can now conclude that the attacker reaches the highly ranked states mainly through an attack from the host ip_2 to the target host ip_4 by exploiting the *rsh_login* vulnerability. Hence, the administrator needs to put an intrusion detection component on the path between ip_2 and ip_4 or stop the *rsh* service between hosts ip_2 and ip_4. Note that these examples are very simple since they are used for illustration purposes only. Given the recent advances in PageRank technology one can expect our approach to scale to much larger systems.

We also implemented the alternative algorithm for ranking states of an Attack Graph based on random simulation described in Section 3.2. We compared the ranks of states obtained using the two algorithms. Note that the modified PageRank algorithm is parameterized by the damping factor d, and the random simulation based ranking algorithm is parameterized by η. For all the examples we considered, both algorithms give the same ordering of states based on their ranks, when $d = \eta = 0.85$. However, the exact values of ranks differ slightly. We observed that when both parameters are decreased simultaneously, the two

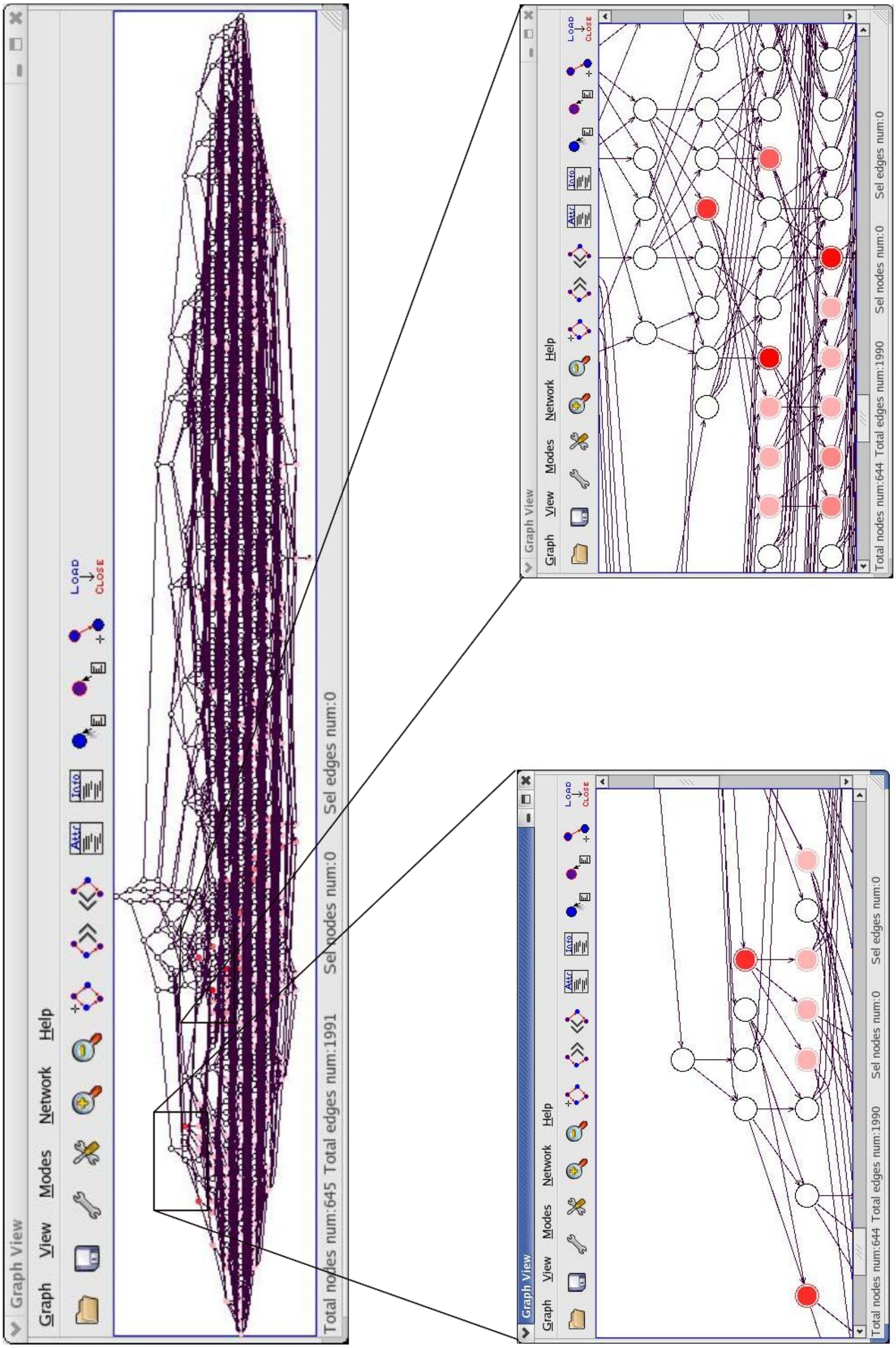

Graph View
Graph
View
Modes
Network
Help
Attr
Info
LOAD
CLOSE
Total nodes num:645 Total edges num:1991 Sel nodes num:0 Sel edges num:0
Total nodes num:644 Total edges num:1990 Sel nodes num:0 Sel edges num:0
Total nodes num:644 Total edges num:1990 Sel nodes num:0 Sel edges num:0

Fig. 5. A large unreadable Attack Graph (left) and a zoom-in of two regions of the Attack Graph with highly ranked states (right)

algorithms still compute the same ordering of ranked states. It is remarkable that two algorithms based on different intuition produce similar results.

6 Conclusions and Future Work

We have given two simple, scalable and useful methods for ranking Attack Graphs. The ranks are a measure of importance of states in an Attack Graph. They provide a metric of security of the system and are useful in making various design decisions aiming at improving security of the system. Ranking helps in overcoming the visual complexity of Attack Graphs by providing a way to view more important areas of the Attack Graph selectively. The first algorithm is similar to Google's PageRank algorithm. The second algorithm computes ranks of states based on the reachability probability of an attacker in a random simulation. Our technique does not assume any knowledge of a priori probabilities for all events. If the probabilities are available, then we can use them for more accurate modeling. Even if the exact probabilities are not available, modeling the attacks randomly is expected to perform as good as PageRank performs on the World Wide Web graph.

A direct extension of this work is to combine the ranks obtained using the above algorithm with other criteria to rank states. Severity of damage occurring at various error states, cost of preventing an error etc. are some other factors which can be used to improve the ranks obtained. Another useful direction for future work is to combine the ranks obtained with logical views of Attack Graph to aid analysis. Noel and Jajodia [18] have given a framework for obtaining hierarchical views of Attack Graphs. The views are obtained using automatic aggregation on common attribute values for elements of the system or connectedness of the Attack Graph. Ranks over aggregated states can be used to get an idea of the probability of attack or damage at various elements in the system.

References

1. Monica Bianchini, Marci Gori, and Franco Scarselli. Inside pagerank. In *ACM Transactions on Internet Technology*, pages 92–128, 2005.
2. Sergey Brin and Larry Page. Anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.
3. E. Clarke, O. Grumberg, and D. Peled. Model checking. In *MIT Press*, 2000.
4. M. Dacier, Y. Deswarte, and M. Kaaniche. Quantitative assessment of operational security: Models and tools. Technical Report 96493, LAAS, May 1996.
5. J. Dawkins and J. Hale. A systematic approach to multi-stage network attack analysis. In *Proceedings of the Second IEEE International Information Assurance Workshop*, 2004.
6. T. Haveliawala. Efficient computation of pagerank. In *Stanford DB Group Technical Report*, 1999.
7. T. Haveliawala, S. Kamvar, and G. Jeh. An analytical comparison of approaches to personalizing pagerank. In *Stanford University Technical Report*, 2003.

8. S. Jha, O. Sheyner, and J. M. Wing. Minimization and reliability analysis of attack graphs. In *CMU CS Technical Report*, Feb 2002.
9. S. Jha, O. Sheyner, and J.M. Wing. Two formal analyses of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, pages 49–63, Nova Scotia, Canada, June 2002.
10. Somesh Jha and Jeannette M. Wing. Survivability analysis of networked systems. In *23rd International Conference on Software Engineering(ICSE'01)*, page 0307, 2001.
11. S. Kamvar, T. Haveliawala, and G. Golub. Adaptive methods for the computation of pagerank. In *Stanford University Technical Report*, 2003.
12. S. Kamvar, T. Haveliawala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. In *Stanford University Technical Report*, 2003.
13. S. Kamvar, T. Haveliawala, C. Manning, and G. Golub. Extrapolation methods for accelerating pagerank computations. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.
14. A. Kuehlmann, K. L. McMilan, and R. K. Brayton. Probabilistic state space search. In *Proceedings of ACM/IEEE international conference on Computer Aided Design*, 1999.
15. Amy N. Langville and Carl D. Meyer. Deeper inside pagerank. In *Internet Mathematics*, pages 335–400, 2004.
16. Chris Pan-Chi Lee, Gene H. Golub, and Stefanos A. Zenios. A fast two-stage algorithm for computing pagerank and its extensions. In *Scientific Computation and Computational Mathematics*, 2003.
17. B. B. Madan, K. G. Popstojanova, K. Vaidyanathan, and K. S. Trivedi. A method for modeling and quantifying the security attributes of intrusion tolerant systems. In *Dependable Systems and Networks-Performance and Dependability Symposium*, pages 167–186, 2004.
18. S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, Washington DC, USA, 2004.
19. R. Ortalo, Y. Deshwarte, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. In *IEEE Transactions on Software Engineering*, pages 633–650, Oct 1999.
20. C.A. Phillips and L. P. Swiler. A graph-based system for network vulnerability analysis. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, pages 71–79, June 2000.
21. O. Sheyner, J.Haines S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002.
22. O. Sheyner and J.M. Wing. Tools for generating and analyzing attack graphs. In *Proceedings of Workshop on Formal Methods for Components and Objects*, pages 344–371, 2004.
23. S. Staniford, V. Paxson, and N. Weaver. How to own the internet in your spare time. In *Proceedings of the 11th USENIX Security symposium*, 2002.
24. HF. Zhu. The methematical models of computer virus infection and methods of prevention. In *Mini-Micro Systems (Journal of China Computer Society)*, pages Vol 11, No.7, 14–21, 1990.
25. Cliff C. Zou, Don Towsley, and Weibo Gong. Email virus and worm propagation simulation. In *13th International conference on Computers Communications and Networks*, Chicago, Oct. 2004.

Appendix

In this section, we prove the existence of a unique probability distribution among all the states after a long run for the Probabilistic Attack Model constructed by us.

Theorem. *A Probabilistic Attack Model constructed by us converges to a unique stationary distribution.*

Proof. The Probabilistic Attack Model can be viewed as a Markov Chain where the probability on each edge is the transition probability. Hence, the rank of a state is actually its limiting probability in Markov theory, which is simply defined as the probability of reaching this state after a long time. Unfortunately, this limiting probability may not always exist, and may not be unique for a general state transition model. Here, we provide a proof that it exists and is also unique for the Probabilistic Attack Model constructed by us and thus our computation converges to the correct ranks.

To prove this, we need to prove that the Probabilistic Attack Model constructed by us is an ergodic Markov chain. If a Markov chain is ergodic, each state will converge to a unique limiting probability. In this case we say that the Markov chain has a unique stationary distribution. In order for the chain to be ergodic, it must satisfy three properties: the chain must be irreducible, positive recurrent and aperiodic. In an irreducible chain any state can be reached from any other state in the graph with probability that is greater than 0. Note that a state can return to itself through different paths, i.e., recurrence paths. The number of steps on these recurrence paths is defined as the *recurrence step* (or recurrence time). The positive recurrence property requires that for any state, the mean recurrence step is finite. Finally, an aperiodic chain requires each state to be aperiodic. A state is called aperiodic if the greatest common divisor of its recurrence steps is 1. If the chain is proved to be ergodic, a well-known theorem in the Markov theory states that the chain will converge to a unique stationary distribution.

Recall that from any state (except the initial state), there is an edge pointing back to the initial state. In our Probabilistic Attack Model, by definition, starting from the initial state, any other state can be reached. On the other hand, each edge has a probability that is greater than 0. This means that all states can reach each other through the initial state with probability that is greater than 0.

Our Probabilistic Attack Model has a finite number of states. Since it is irreducible, a state is always able to return to itself through the initial state in a finite number of steps. A result in Markov theory shows that the mean recurrence step of each state is finite. Thus, this chain is positive recurrent.

In order to prove our model is ergodic, it remains to prove the chain is aperiodic. We define a state to be *dangling* if there is no other outgoing edge from it except for the edge that points back to the initial state. The aperiodicity proof is divided into two cases in Figure 6.

In case 1, all the successive states j of the initial state are dangling. The recurrence step for each state is $2, 4, 6, \ldots$, thus all the states are periodic and

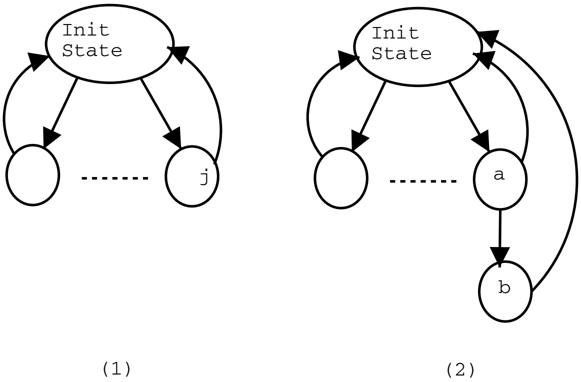


Fig. 6. Two Cases in the Probabilistic Attack Model

the chain is periodic. Note that this case is trivial and does not appear in practice. However, to make this chain aperiodic is easy: a self loop can be added to the initial state, with a tiny probability ϵ. Thus, we simply ignore this case.

In case 2, the initial state has at least one non-dangling successor: state a whose successor b is a dangling state. State b can be shown to have a 3-step and 5-step recurrence. The 3-step recurrence is completed by moving from state b to the initial state and back to b through a. The 5-step recurrence happens when the systems goes from state b to the initial state, and then back to the initial state through a, and finally back to b. The $gcd(3, 5) = 1$ therefore state b is aperiodic. Since the chain is irreducible, all states reach each other. A theorem in Markov theory shows that these states have the same periods. Thus, all the states are aperiodic.

Since the chain is irreducible, positive recurrent and aperiodic, it is ergodic. Hence, it has a unique stationary distribution that can be computed through our modified PageRank algorithm. Q.E.D.