Securing Networks against Unpatchable and Unknown Vulnerabilities Using Heterogeneous Hardening Options

Daniel Borbor[1], Lingyu Wang[1], Sushil Jajodia[2], and Anoop Singhal[3]

[1] Concordia Institute for Information Systems Engineering, Concordia University
{d_borbor,wang}@ciise.concordia.ca
[2] Center for Secure Information Systems, George Mason University
jajodia@gmu.edu
[3] Computer Security Division, National Institute of Standards and Technology
anoop.singhal@nist.gov

Abstract. The administrators of a mission critical network usually have to worry about non-traditional threats, e.g., how to live with known, but unpatchable vulnerabilities, and how to improve the network's resilience against potentially unknown vulnerabilities. To this end, network hardening is a well known preventive security solution that aims to improve network security by taking proactive actions, namely, hardening options. However, most existing network hardening approaches rely on a single hardening option, such as disabling unnecessary services, which becomes less effective when it comes to dealing with unknown and unpatchable vulnerabilities. There lacks a heterogeneous approach that can combine different hardening options in an optimal way to deal with both unknown and unpatchable vulnerabilities. In this paper, we propose such an approach by unifying multiple hardening options, such as firewall rule modification, disabling services, service diversification, and access control, under the same model. We then apply security metrics designed for evaluating network resilience against unknown and unpatchable vulnerabilities, and consequently derive optimal hardening solutions that maximize security under given cost constraints.

1 Introduction

Today's computing networks are playing the role of nerve systems in many mission critical infrastructures, such as cloud data centers and smart grids. However, the scale and severity of security breaches in such networks have continued to grow at an ever-increasing pace, which is evidenced by many high profile security incidents, such as the recent large scale DDoS attacks caused by the Mirai Botnet on the Dyn DNS, and the cyber-physical attack on Ukraine power grid in 2015. The so-called zero day attacks, which exploit either previously unknown or known, but unpatched vulnerabilities, are usually behind such security incidents, e.g., Stuxnet employs four different zero day vulnerabilities to target SCADA. Therefore, administrators of a mission critical network usually need to worry about not only patching known vulnerabilities and deploying traditional defense mechanisms (e.g., firewalls, IDSs, and IPSs), but also non-traditional security threats, e.g., how to live with known, but unpatchable vulnerabilities, and how to improve the network's resilience against potentially unknown vulnerabilities.

In fact, it is well known that both cybercriminals and governmental agencies stockpile vulnerabilities that are not publicly known (e.g., the NSA reportedly spent more than 25 million a year to acquire software vulnerabilities, and private vendors are providing at least 85 zero-day exploits on any given day [17]). On the other hand, even for known vulnerabilities, patching is not always a viable option. For example, a patch may not be readily available at the time of the attack, or the system may have reached their end-of-support with no more patch available; patching a vulnerability may cause unacceptable service disruptions on a regular basis (e.g., Windows updates); even worse, patching a vulnerability may sometimes reintroduce other security vulnerabilities that have previously been fixed (e.g., Apache MINA SSHD 2.0.14 introduces an SSL regression previously fixed in 2.0.13 [21]).

Consequently, security professionals need to block the exploitation of such vulnerabilities through other means, such as modifying firewall rules, service diversification, or access control. A critical question is *How to optimally combine such options in order to both improve the security and lower the cost?* To this end, network hardening is a well known preventive security solution that aims to improve network security by taking proactive actions, namely, hardening options. However, most existing network hardening approaches rely on a single hardening option, such as disabling unnecessary services [9, 22] or service diversification [6] (a detailed review of related work will be given later in Section 5). Such a solution becomes less effective when it comes to dealing with unknown and unpatchable vulnerabilities. There lacks a heterogeneous approach that can combine different hardening options in an optimal way to deal with such vulnerabilities.

Running Example We first consider a concrete example to demonstrate why deriving an optimal hardening solution with heterogeneous hardening options would demand a systematic and automated approach. Figure 1 shows a hypothetical network roughly based on Cisco's cloud data center concept [5] as well as the OpenStack architecture [12]. Despite its relatively small scale, it mimics a typical cloud network, e.g., the client layer connects the cloud network to the internet through the CRS 7600; a firewall (ASA v1000) separates the outside network from the inner one. There is a security/authentication layer (authentication server, Neutron server, etc.) as well as a VM and Application layer (Web and application servers). Finally, a storage layer is separated and protected by another firewall (ASA 5500) and an MDS 9000.

We make following assumptions about the network. We assume the two firewalls and other host-based security mechanisms (e.g., personal firewalls or iptables) together enforce the connectivity described inside the connectivity table shown in the figure. External users (including attackers) are represented with host $h0$, and the most critical asset is assumed to be the Xen database server ($h4$), which may be accessed through the three-tier architecture involving hosts $h1$, $h2$, and $h3$. We assume the network is free of any known vulnerabilities, except for an unpatchable vulnerability on the application server running SecurityCenter 5.5 (which cannot be changed due to functionality requirements), and another one on the database server running MySQL 5.7 which may be changed to MSQL 2012 or PostgreSQL 9. For simplicity, we exclude exploits and conditions that involve firewalls in this example.

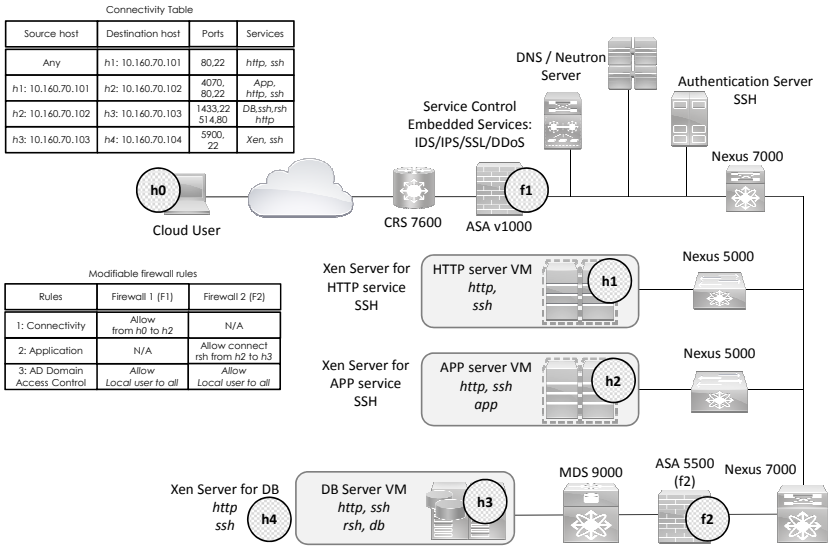

Fig. 1. An Example Cloud Network.

To measure the network's resilience against zero-day attacks, we apply the *k-zero-day safety metric* ($k0d$) [26]. This metric basically counts how many distinct services must be compromised using unknown vulnerabilites before an attacker may compromise the critical asset (i.e., the number of distinct services along the shortest path). In addition, we refine the metric by taking into consideration the potentially uneven distribution of distinct services along the shortest path [30, 33] (e.g., a path consisting of three $http$ and one Xen would be considered slightly "shorter", or less secure, than a path consisting of two $http$ and two Xen, although both paths have the same number of resource instances and resource types).

For hardening options, we consider changes of both the firewall rules and service types. First, we assume the administrator may enable or disable firewall rules on both the firewall ASA v1000 ($f1$) and on the firewall ASA 5500 ($f2$). On $f1$ he has a rule that allows the connection from the cloud user ($h0$) to the app VM ($h2$); he also has the option to allow local user access to $h1$ and $h2$. The firewall ASA 5500 ($f2$) has a rule where he allows the rsh connection on $h3$ from $h2$, as well as local user access to $h3$ and $h4$. Second, we assume the administrator has the option of replacing the Apache Mina 2.0.14 ssh servers with either Copssh 5.8, OpenSSH 7.4, or Attachmate 8.0; the Web servers with either Apache 2.4, IIS 8.5, NGINX 1.9 or a Litespeed 5.0.14 Web server; the rsh service only uses MVRSHD 2.2.

Clearly, even with such a small scale network, the administrator now faces a number of hardening options, including disabling service instances, diversifying service types, and changing firewall rules, each of which may incur certain installation/maintenance cost (we will discuss the cost model in more details later in Section 2). To maximize the resilience of the network against both unknown and unpatchable vulnerabilities, the administrator must decide what would be the optimal combination of such harden-

ing options in order to maximize the aforementioned security metric, while respecting given cost constraints. Such a task would obviously be tedious and error prone, if done manually, and demands a systematic and automated approach.

In this paper, we develop such an approach to optimally combine heterogeneous hardening options in order to increase a network's resilience again both unknown and unpatchable vulnerabilities under various cost constraints. Specifically, we first devise our model of different hardening options, costs, and the security metric. We then develop optimization and heuristic algorithms to derive optimal hardening solutions under given cost constraints. We evaluate our approach through simulations in order to study the effect of optimization parameters on accuracy and running time, and the effectiveness of optimization for different types of networks. In summary, the main contribution of this paper is the following.

- To the best of our knowledge, this is the first effort on network hardening using heterogeneous hardening options.
- In constrast to previous works, we provide a refined security metric and an improved cost model that takes into account real world variables in calculating hardening costs.
- Our method is practically relevant to the defense of mission critical networks in which unknown and unpatchable vulnerabilities are realistic security concerns.

The remainder of this paper is organized as follows: In Section 2, we present the model and formulate the optimization problem, and in Section 3 we discuss the methodology and show case studies. Section 4 shows simulation results. Section 5 reviews related work and Section 6 concludes the paper.

2 The Model

We first introduce the extended resource graph model to capture network services and their relationships, then we present the heterogeneous hardening control and cost model, followed by problem formulation.

2.1 Extended Resource Graph

To model network services and their relationships, we revise the *Extended Resource Graph* concept introduced in our previous work [6] in order to model both unpatchable and unknown vulnerabilities, as well as heterogeneous hardening options. The extended resource graph of the running example is shown in Figure 2 and detailed below.

In Figure 2, each pair shown in a rectangle is a security-related condition. If the condition is a privilege, it is represented as $\langle privilege, host \rangle$; if it is connectivity, it is represented as $\langle source, destination \rangle$. If a firewall affects a security-related condition, it is represented as $\langle privilege, firewall, host \rangle$ or as $\langle source, firewall, destination \rangle$. Each one of the rows below the rectangle indicate different hardening options available for that condition. The option currently in use is indicated by the highlighted integer (e.g., 0 means disabled; in the case of service diversification, 1 means Apache, and 2

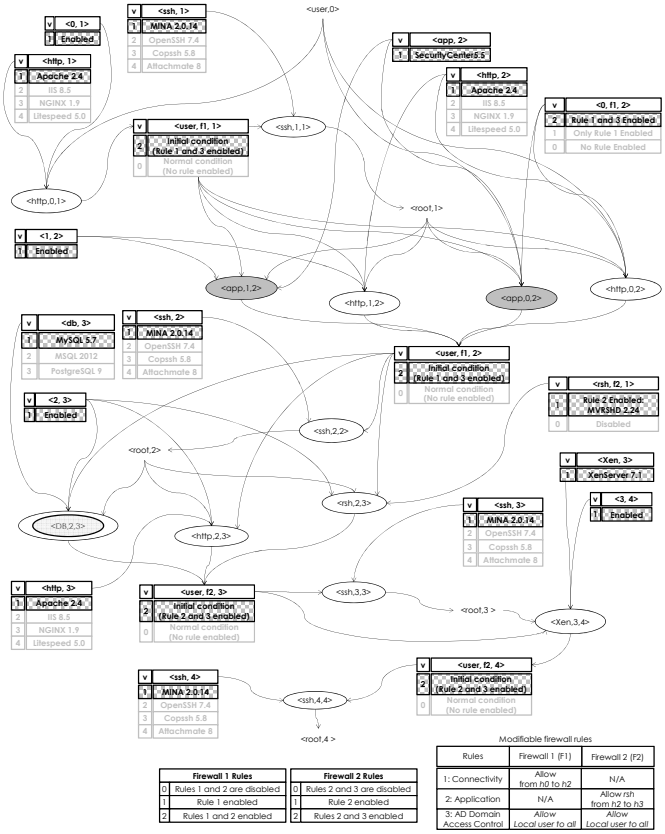

Fig. 2. The extended resource graph of our running example.

means IIS) and other potential instances are in a lighter text. For the conditions modifiable by a firewall rule, the rows below the rectangle indicate the firewall rules that affect it.

Each exploit node (oval) is a tuple that consists of a service running on a destination host, the source host, and the destination host (e.g., the tuple $\langle http, 1, 2\rangle$ indicates a potential zero-day vulnerability in the $http$ service on host 2, which is exploitable from host 1). If the exploit is unpatchable, but diversifiable, it is represented by a double oval; if it is neither patchable nor diversifiable, it is represented as a colored oval (those different types of exploits will contribute to the calculation of the security metric value, as detailed later). The self-explanatory edges point from preconditions to an exploit (e.g., from $\langle 0, 1\rangle$ and $\langle http, 1\rangle$ to $\langle http, 0, 1\rangle$), and from the exploit to its post-conditions (e.g., from $\langle http, 0, 1\rangle$ to $\langle user, 1\rangle$).

We make two design choices here. The first is to associate the service instance concept as a property (label) of a condition (e.g., $\langle http, 1\rangle$), instead of an exploit (as in our previous work [6]). This label can then be inherited by the corresponding exploits. The second design choice is that, while some conditions indicate the involved firewall rules, the actual label values that they will take will depend on the number of predefined modifiable rules in the firewall itself. For each firewall, instead of modeling service instances,

we model the number of modifiable firewall rules that can be enabled. This would help to avoid the need for introducing new conditions and exploits into the extended resource graph when firewall rules are to be disabled and hence we may work with a fixed structure of the extended resource graph. While the definitions of service pool and service instance remain the same as in [6], Definitions 1 and 2 formally introduce the revised concepts.

Definition 1 (Firewall Rule Pool and Firewall Rule). *Denote F the set of all firewalls and Z the set of integers, for each firewall $f \in F$, the function $r(.) : F \to \mathsf{Z}$ gives the firewall rule pool of f which represent all modifiable firewall rules of that firewall.*

Definition 2 (Extended Resource Graph). *Given a network composed of*

- *a set of hosts H,*
- *a set of services S, with the service mapping $serv(.) : H \to 2^S$,*
- *the collection of service pools $SP = \{sp(s) \mid s \in S\}$,*
- *the collection of firewall rules $FR = \{r(f) \mid f \in F\}$,*
- *a set of firewalls F, with the rule mapping $r(.) : F \to \mid FR \mid$,*
- *and the labelling function $v(.) = v_f(.) \cup v_c(.)$ where $v_f(.) : f \to F$ and $v_c(.) : C \to SP$*

Let E be the set of zero day exploits $\{\langle s, h_s, h_d\rangle \mid h_s \in H, h_d \in H, s \in serv(h_d)\}$, and $R_r \subseteq C \times E$ and $R_i \subseteq E \times C$ be the collection of pre and post-conditions in C. We call the labeled directed graph, $\langle G(E \cup C, R_r \cup R_i), v\rangle$ the extended resource graph.

2.2 Heterogeneous Hardening Control and Cost Model

We introduce the notion of *heterogeneous hardening control* as a model to account for all hardening options in a network where we represent each initial condition as an optimization variable. We formulate the heterogeneous hardening control vectors using those variables as follows. We note that the number of optimization variables present in a network will depend on the number of initial conditions that are affected by one or more hardening options. Since we only consider remotely accessible services in the extended resource graph model, we would expect in practice the number of optimization variables to grow linearly in the size of the network (i.e., the number of hosts).

Definition 3 (Optimization Variable and Heteregeneous Hardening Control). *Given an extended resource graph $\langle G, v\rangle$, $\forall c \in C$ and $\forall f \in F$, $v(c)$ and $v(f)$ are optimization variables. A hardening control vector is the integer valued vector $\boldsymbol{V} = (v(c_1), v(c_2), ..., v(c_{|C|}) \cup (v(f_1), v(f_2), ..., v(f_{|F|})$*

Changing the value of an optimization variable has an associated *hardening cost* and the collection of such costs are given in a *hardening cost matrix* in a self-explanatory manner. We make use of Gartner's 2003 *Total Cost of Ownership* (TCO) analysis report [20] to establish a realistic cost estimation of the cost of different hardening options. Table 1 provides a reference as to which criteria is applicable to different hardening options costs.

Hardening Option Cost Selection Criteria				
Gartner's TCO criteria	Firewall Connectivity	Firewall Layer 3	Firewall Access Control	Diversity
Downtime Costs				x
Operational Costs	x	x	x	x
Support Costs	x		x	x
Changes in upgrade Costs	x	x	x	x
Monitoring costs			x	x
Production costs				x
Security management and failure control costs	x	x	x	x

Table 1. Criteria to be used when calculating hardening costs for different hardening options based on Gartner's TCO [20]

Definition 4 (Hardening Cost). *Given $s \in S$ and $sp(s)$, and given $f \in F$ and $r(f)$, the cost to change from one specific hardening option to another is defined as the hardening cost.*

Definition 5 (Hardening Cost Matrix). *The collection of all hardening costs for all hardening options are given as a hardening cost matrix HCM. For the different hardening options, the element at i^{th} row and j^{th} column indicates the hardening cost of changing the i^{th} hardening option to be the j^{th} hardening option.*

Definition 6 (Total Hardening Cost). *Let $v_s(c_i)$ be the service associated with the optimization variable $v(c_i)$ and $\boldsymbol{V}_{c0}$ the initial service instance values for each of the conditions in the network. Let $v_f(f_i)$ be the firewall associated with the optimization variable $v(f_i)$ and $\boldsymbol{V}_{f0}$ the initial firewall rule set values for each of the firewalls in the network. The total hardening cost, Q_d, given by the heterogeneous hardening vector $\boldsymbol{V}$ is obtained by*

$$Q_d = \sum_{i=1}^{|C|} CM_{v_s(c_i)}(\boldsymbol{V}_{c0}(i), \boldsymbol{V_c}(i)) + \sum_{i=1}^{|F|} CM_{v_f(f_i)}(\boldsymbol{V}_{f0}(i), \boldsymbol{V_f}(i))$$

We note that the above definition of hardening cost between each pair of service instances has some advantages. For example, in practice we can easily imagine cases where the cost is not symmetric, i.e., changing one service instance to another (e.g., from Apache to IIS) carries a cost that is not necessarily the same as the cost of changing it back (from IIS to Apache). Our approach of using a collection of two-dimensional matrices allows us to account for cases like this. Additionally, by considering instance 0, it provides us the advantage to model disabling a service as a special case of service diversification, if the hardening option allows it.

2.3 Problem formulation

As mentioned in Section 1, the security metric that we will be using, denoted as d, is based on the minimum number of distinct resources, excluding those with unpatchable vulnerabilities, on the shortest attack path in the resource graph [26], with the extension for considering the uneven distribution of services along that path [30, 33], as formally defined below.

Definition 7 (d-Safety Metric). *Given an extended resource graph $\langle G(E \cup C, R_r \cup R_i), v\rangle$, and a goal condition $c_g \in C$; let $t = \sum_{i=1}^{n} 2^{-n} \mid serv(h_i)) \mid$ (total number of service instances), and let $p_j = \frac{|h_i : s_j \in serv(h_i))|}{t}$ ($1 \leq i \leq n$, $1 \leq j \leq n$) (relative frequency of each resource). For each $c \in C$ and $q \in seq(c)$ (attack path), denote $R(q)$ for $s : s \in$ R, r appears in q, r is not unpatchable, we define the network's d-safety metric (where min(.) returns the minimum value in a set) $d = min_{q \in seq(c_g)} r(R(q))$; where $r(R(q))$ is the attack path's effective richness of the services, defined as $r(G) = \frac{1}{\prod_1^n p_i^{p_i}}$ [30]*

With the aforementioned models, the network hardening problem is to maximize the d value by changing the hardening options while respecting the available budget in terms of given cost constraints. In the following, we formally formulate this as an optimization problem.

Problem 1 (d-Optimization Problem). Given an extended resource graph $\langle G, v\rangle$, find a heterogeneous hardening control vector $\boldsymbol{V}$ which maximizes $min(d(\langle G(\boldsymbol{V}), v\rangle))$ subject to the constraint $Q \leq B$, where B is the available budget and Q is the total hardening cost as given in Definition 6.

Since our problem formulation is based on an extended version of the resource graph, which is syntactically equivalent to attack graphs, many existing tools developed for the latter (e.g., the tool in [16] has seen many real applications to enterprise networks) may be easily extended to generate extended resource graphs which we need as inputs. Additionally, our problem formulation assumes a very general model of budget B and cost Q, which allows us to account for different types of budgets and cost constraints that an administrator might encounter in practice, as will be demonstrated in the following section.

3 The Methodology

This section details our optimization and heuristic algorithms used for solving the formulated heterogeneous hardening problem. We also illustrate the optimization process through a few case studies.

3.1 Optimization Algorithm

Our first task is to select an optimization algorithm that would fit our hardening problem. First, it is well known that most gradient-based methods require to satisfy mathematical properties like convexity or differentiability, which are not applicable to our

problem. Second, the problem we want to solve includes different if-then-else constructs to account for the different hardening technique used, and thus, an algorithm that allows to insert this construct is necessary. Additionally, since our optimization problem uses variables that are defined as discrete (discrete variable space), a simple and robust search method and optimization technique is needed. We find that metaheuristic algorithms provide these advantages. Specifically, the Genetic Algorithm (GA) provides a simple and clever way to encode candidate solutions to the problem [8]. One of the main advantages is that we do not have to worry about explicit mathematical definitions. For our automated optimization approach, we chose GA because it requires little information to search effectively in a large search space in contrast to other optimization methods (e.g., the mixed integer programming [4]).

The extended resource graph is the input to our automated optimization algorithm where the fitness function is d. One important point to consider when optimizing the d function on the extended resource graph is that, for each generation of the GA, the graph's labels selected will dynamically change. This in turn will change the value of d, since the shortest path may have changed with each successive generation of GA and the change in the hardening options will enable or disable certain paths. Our optimization tool takes this into consideration. Additionally, if there are more than one shortest path that provides the optimized d, our optimization tool gives priority to the paths by considering the uneven distribution and relative frequency of resources in that path, thus addressing one of the limitations that was present in [6] where no priority was provided.

The constraints are defined as a set of inequalities in the form of $q \leq b$, where q represents one or more constraint conditions and b represents one or more budgets. These constraint conditions can be overall constraints (e.g., the total hardening cost Q_d) or specific constraints to address certain requirements or priorities while implementing the heterogeneous hardening options. The number of independent variables used by the GA (genes) are the optimization variables given by the extended resource graph. For our particular network hardening problem, the GA will be dealing with integer variables representing the selection of a hardening option. Because $v(.)$ is defined as an integer, the optimization variables need to be given a minimum value and a maximum value. This range is determined by the number of instances provided in the service pool of each service and firewall rule pool of each firewall. The initial service instance for each of the services and the initial set of firewall rules are given by the extended resource graph while the final heterogeneous hardening control vector $\boldsymbol{V}$ is obtained after running the GA.

3.2 Case Studies

In the following, we demonstrate different use cases of our method with varying cost constraints and hardening options. For these test cases, the population size defined for our tool is set to be at least the value of optimization variables (more details will be provided in the coming section). This way we ensure the individuals in each population span the search space. We ensure the population diversity by testing with different settings in genetic operations (like crossover and mutation). For all the test cases, we have used the following algorithm parameters: population size = 100, number of generations = 150, crossover probability = 0.8, and mutation probability = 0.2.

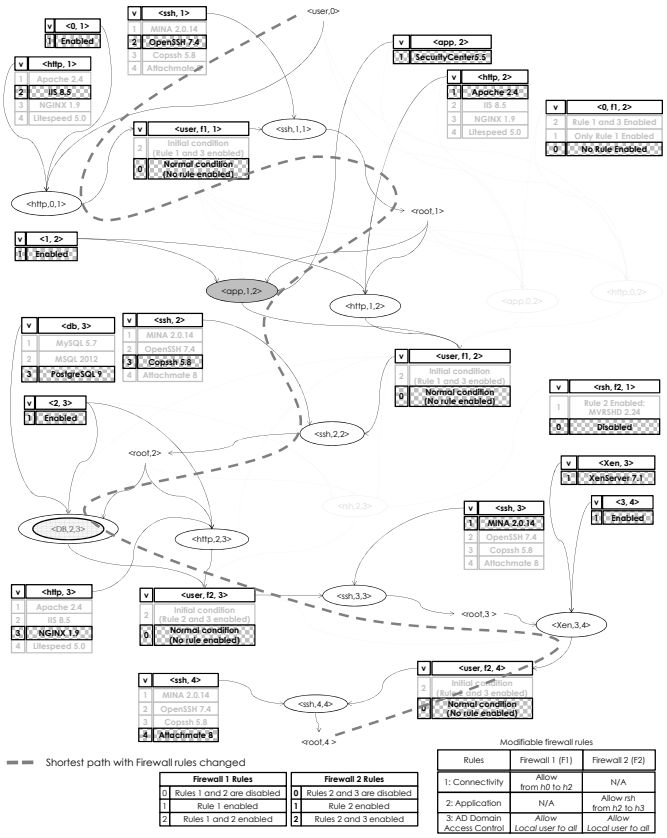

Fig. 3. Test case A: Effect of modifiable hardening options and budget constraints.

Test case A: $Q_d \leq 500$ units with firewall rule constraints. We start with the simple case of one overall budget constraint ($Q_d \leq 500$). There are 11 different services-based optimization variables and 2 firewall-based optimization variables. If no firewall rules are changed, the solution provided by the GA yields d=2.7529. In this case, because of the firewall rules that are enabled, the metric cannot be increased any further.

On the other hand, if we allow the firewall rules to be modified, while maintaining the overall budget $Q_d \leq 500$, the optimization results will be quite different. The solution provided by the GA is a d metric of 3.3895. This total hardening cost satisfies both the overall budget constraints. We can see that the hardening options enforced by the firewall rules in our optimization tool can affect the optimization. Nevertheless, additional budget constraints might not allow achieving the maximum d possible.

Test case B: $Q_d \leq 500$ units with a critical service with an unpatched vulnerability. While test case A shows how enabling or disabling predefined firewall rules can affect the d metric optimization, when considering the effects of unpatchable vulnerabilities the d metric value will change. This test case models such a scenario by assigning a restriction for the ssh services not to be diversified or disabled.

In the graph, we can see that the ssh service is highlighted to represent the fact that it cannot be patched. The solution provided by the GA is d=2.8284. While the increase

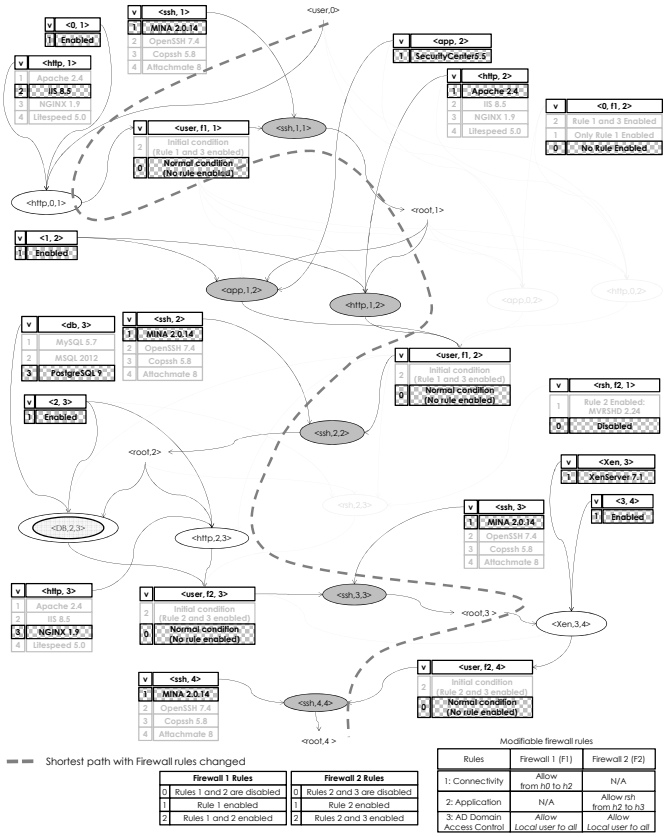

Fig. 4. Test case B: Effect of having an unpatchable vulnerability in the network.

is less than when the ssh service can be diversified, we can still have an increase in the d metric even with unpatchable vulnerabilities on the network.

As seen from the above test cases, our model and problem formulation makes it relatively straightforward to apply any standard optimization techniques, such as the GA, to optimize the d metric through combining different network hardening options while dealing with unpatchable and unknown vulnerabilities and respecting given cost constraints.

3.3 Heuristic Algorithm

All the test cases described above rely on the assumption that all the attack paths are readily available. However, this is not always the case in practice. Due to the well-known complexity that resource graphs have inherited from attack graphs due to their common syntax [30, 33], it is usually computationally infeasible to enumerate all the available attack paths in a resource graph for large networks. Therefore, we present a modified version of the heuristic algorithm [6] to reduce the search complexity when calculating and optimizing the d metric by only storing the m-shortest paths at each step. The following briefly describes the modified algorithm.

The algorithm starts by finding the initial conditions that are affected by the modifiable firewall rules and stores them on a list γ. After that, it topologically sorts the graph and proceeds to go through each one of the nodes on the resource graph. If an exploit is a post-condition of one of the conditions in γ, it is not included in the set of exploits $\sigma()$. The main loop cycles through each unprocessed node. If a node is an initial conditions, the algorithm assumes that the node itself is the only path to it and it marks it as processed. For each exploit e, all of its preconditions are placed in a set. The collection of attack paths $\alpha(e)$ is constructed from the attack paths of those preconditions. In a similar way, $\sigma'(ov(e))$ is constructed with the function $ov()$ which, aside from using the exploits, includes value of elements of the hardening control vector that supervises that exploit.

If there are more than m paths to that node, the algorithm will first look for the relative frequency of each unique combination of exploit and service instance in $\alpha'(ov(e))$. Then, the algorithm creates a dictionary structure where the key is a path from $\alpha(e)$ and the value is the effective richness of service/service instance combinations given by each one of the respective paths in $\alpha'(ov(e))$. A function $ShortestM()$ selects the top m keys whose values are the smallest and returns the m paths with the smallest effective richness value. If there are less than m paths, it will return all of the paths. After this, it marks the node as processed. The process is similar when going through each one of the intermediate conditions. Finally, the algorithm returns the collection of m paths that can reach the goal condition c_g. It is worth noting that by considering the effective richness of each path, the algorithm provides gives a path a priority based on the relative frequency of the combination of unique service with service instance.

4 Simulations

In this section, we show simulation results. All simulations are performed using a computer equipped with a 3.0 GHz CPU and 8GB RAM in the Python 2.7.10 environment under Ubuntu 12.04 LTS and MATLAB 2015a's GA toolbox. To generate a large number of resource graphs for simulations, we first construct a small number of seed graphs based on realistic cloud networks and then generate larger graphs from those seed graphs by injecting new hosts and assigning resources in a random but realistic fashion (e.g., the number of pre-conditions of each exploit is varied within a small range since real world exploits usually have a constant number of pre-conditions).

For the different hardening options that are implemented through firewall rules, we randomly select 10% of the initial conditions. Additionally, to analyze the effect of unpatchable vulnerabilities, our graphs include randomly assigned unpatchable services. The resource graphs are used as the input for the optimization toolbox where the objective function is to maximize the minimum d value subject to budget constraints. In all the simulations, we employ the heuristic algorithm described in section 3.3.

To determine the genetic operators, we used the hill climbing algorithm. Our simulations showed that (detailed simulation results are omitted here due to page limitations), using the GA with a crossover probability of 80%, a mutation rate of 20%, and setting the number of generations to 70 will be sufficient to obtain good results. Additionally, our experiences also show that, because our largest resource graph had a heterogeneous

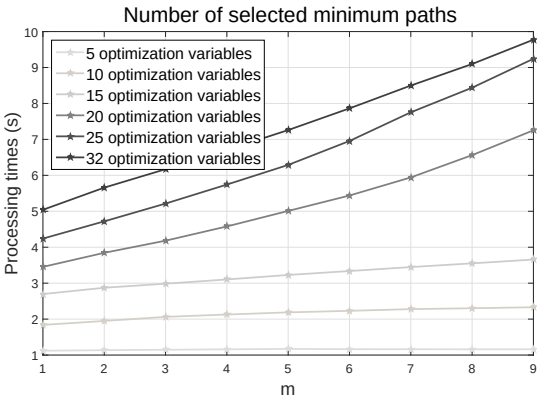

Fig. 5. Processing time.

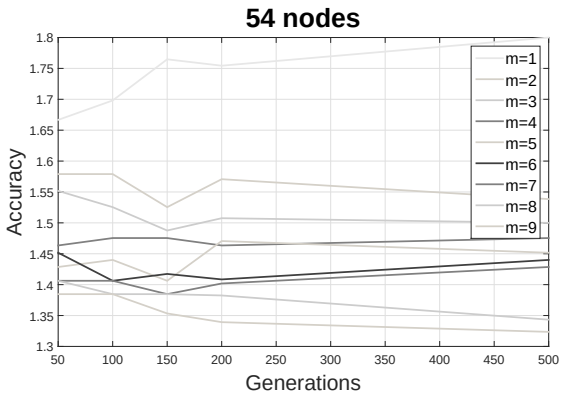

Fig. 6. Accuracy vs m (parameter of the heuristic algorithm).

hardening control vector of less than 100 variables, we could set the population size equal to 200; nevertheless, we believe that when dealing with a bigger number of optimization variables, the population size should be at least twice the number of variables.

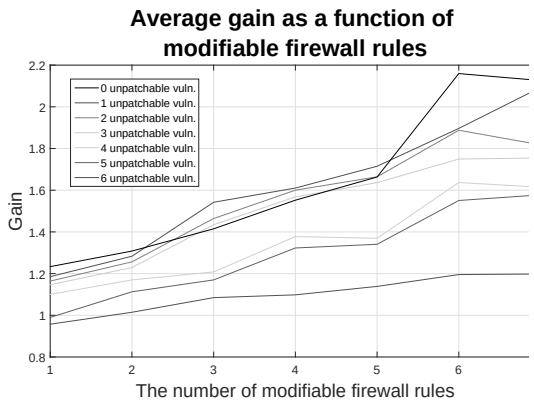

Fig. 7. The average gain based on the number of modifiable firewall rules.

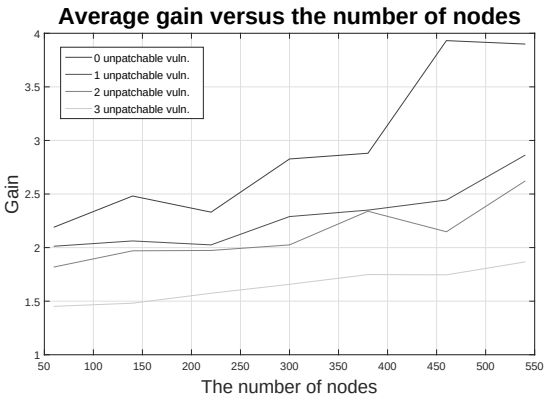

Fig. 8. The average gain vs the number of nodes.

Figure 5 shows that the processing time increases almost linearly as we increase the number of optimization variables or the parameter m of the heuristic algorithm. The results show that the algorithm is relatively scalable with a linear processing time. On the other hand, the accuracy of the results is also an important issue to be considered. Here the accuracy refers to the approximation ratio between the result obtained using the heuristic algorithm and that of the brute force algorithm (i.e., simply enumerating and searching all the paths while assuming all services and service instances are dif-

ferent). For the simulations depicted in Figure 6, we settled for 50 iterations per graph per m-paths. The heterogeneous hardening control vector provided by the GA is used to calculate the accuracy. From the results, we can see that when m is greater or equal to 4 the approximation ratio reaches an acceptable level. For the following simulations, we have settled with an m value of 6.

Figure 7 shows that the gain will increase linearly as we increase the number of firewall-based hardening options. These results confirm that firewall-based hardening options can positively affect our effort to provide better resilience for cloud networks against zero-day attacks. Additionally, the figure shows that the number of unpatchable vulnerabilities that are present in the network will significantly reduce the gain that can be achieved through other hardening techniques. Since it is not probable to find a large number of unpatchable vulnerabilities all at the same time within a network, we only consider up to three unpatchable vulnerabilities.

In Figure 8, we analyze the average gain in the optimized results for different sizes of graphs. In this figure, we can see that we have a good enough gain for graphs with a relatively high number of nodes. As expected, as we increase the number of unpatchable vulnerabilities, the gain will decrease. However, we can also see this decrease is linear. For those simulations, we have used a population size of 300, 50 generations, and a crossover fraction of 80%.

Figures 9 and 10 show the optimization results of different shapes of resource graphs in terms of depth and degree of exposure, which roughly represents the extent to which the network is protected. While it may be difficult to exactly define the depth of a resource graph, we have relied on the relative distance, i.e., the difference of the shortest path before and after all hardening options have been applied. There is a linear increase in the gain as we increase the relative distance in the shortest path. This is independent of the amount of unpatchable vulnerabilities. While this does not provide an accurate description of the graph's shape, it does provide an idea of how much our algorithm can increase the minimum d for graphs with different depths.

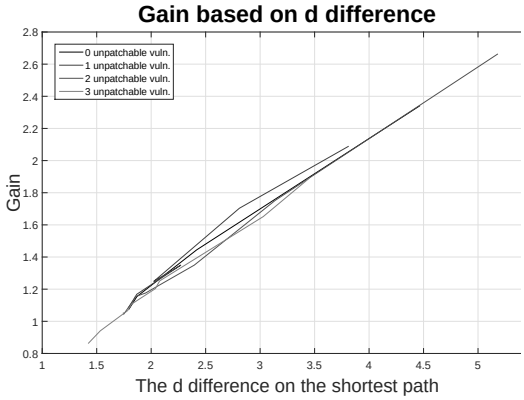

Fig. 9. The d difference on the shortest path.

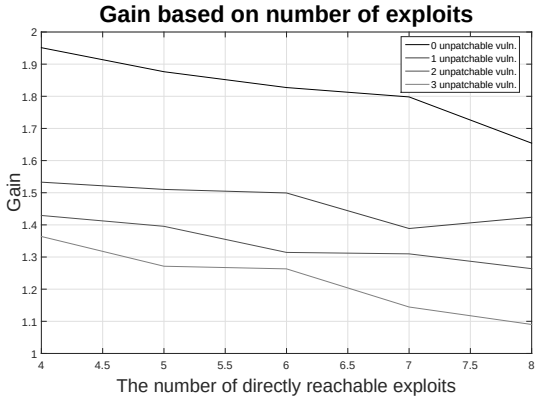

Fig. 10. The number of directly reachable exploits.

Finally, as shown in Figure 10, we can see the effect of the network's degree of exposure, which is defined as the number of exploits that are directly reachable by the attacker from the external host *h0*. As we increase the degree of exposure, the gain in optimization decreases (circles in the graph). That is, there will less room for hardening if the network is more exposed.

5 Related Work

In general, the security of networks may be qualitatively modeled using attack trees [9, 10, 23] or attack graphs [2, 24]. A majority of existing quantitative models of network security focus on known attacks [29, 1], while few works have tackled zero day attacks [27, 26, 30, 33] which are usually considered unmeasurable due to the uncertainties involved [18]. In terms of security metrics, most of the current works deal with assigning numeric scores to rank known vulnerabilities (mostly based on the CVSS) [19] to be able to model the impact that they have on a network. This ranking is based on how likely and easily exploitable the known vulnerabilities are. This, however, is not the case for unknown vulnerabilities.

Early works on network hardening typically rely on qualitative models while improving the security of a network [24, 28, 25]. Those works secure a network by breaking all the attack paths that an attacker can follow to compromise an asset, either in the middle of the paths or at the beginning (disabling initial conditions). Also, those works do not consider the implications when dealing with budget constraints nor include cost assignments, and tend to leave that as a separate task for the network administrators. While more recent works [1, 32] generally provide a cost model to deal with budget constraints, one of the first attempts to systematically address this issue is by Gupta et al. [15]. The authors employed genetic algorithms to solve the problem of choosing the best set of security hardening options while reducing costs.

Dewri et a. [9] build on top of Gupta's work to address the network hardening problem using a more systematic approach. They start by analyzing the problem as a single objective optimization problem and then consider multiple objectives at the same time. Their work consider the damage of compromising any node in the cost model in order to determine the most cost-effective hardening solution. Later on, in [10] and in [31], the authors extrapolate the network hardening optimization problem as vulnerability analysis with the cost/benefit assessment, and risk assessment respectively. In [22] Poolsappasit et al. extend Dewri's model to also take into account dynamic conditions (conditions that may change or emerge while the model is running) by using Bayesian attack graphs in order to consider the likelihood of an attack. Unlike our work, most existing work is limited to known vulnerabilities and focus on disabling existing services.

There exist a rich literature on employing diversity for security purposes. The idea of using design diversity for tolerating faults has been investigated for a long time, such as the N-version programming approach [3], and similar ideas have been employed for preventing security attacks, such as the N-Variant system [7], and the behavioral distance approach [13]. In addition to design diversity and generated diversity, recent work employ opportunistic diversity which already exists among different software systems. For example, the practicality of employing OS diversity for intrusion tolerance is

evaluated in [14]. More recently, the authors in [30, 33] adapted biodiversity metrics to networks and lift the diversity metrics to the network level. While those works on diversity provide motivation and useful models, they do not directly provide a systematic solution for improving diversity. So far, the work done by [6], is one of the first work that has tried to provide a solution for this problem; their limitation, however, is that their metric is too simplistic and does not consider additional hardening options.

6 Conclusions

In this paper, we have provided a heterogeneous approach to network hardening to increase the resilience of a network against both unknown and unpatchable vulnerabilities. By unifying different hardening options within the same model, we derived a more general method than most existing efforts that rely on a single hardening option. Our automated approach employed a heuristic algorithm that helped to manage the complexity of evaluating the security metric as well as limiting the time for optimization to an acceptable level. We have addressed one limitation of our previous work by considering the uneven distribution of services along an attack path. We have devised a more realistic cost model. We have tested the efficiency and accuracy of the proposed algorithms through simulation results, and we have also discussed how the gain in the d value will be affected by the number of available modifiable firewall rules, unpatchable vulnerabilities, and the different sizes and shapes of the resource graphs.

The following lists several future direction of our approach.

- While this paper has proven that we can integrate different network hardening options (e.g., firewalls and diversity) under the same model, some hardening options may not easily fit into this model (e.g., service relocation).
- The security metric we applied relies on the number of unknown vulnerabilities, which may be refined by further considering known and patchable vulnerabilities (even though those would carry less weight).
- This study relies on a static network configuration. A future research direction would be to consider a dynamic network model in which both attackers and defenders may cause incremental changes in the network.
- We will evaluate other optimization algorithms in addition to GA to find the most efficient solution for our problem.

Disclaimer Commercial products are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose.

References

1. Massimiliano Albanese, Sushil Jajodia, and Steven Noel. Time-efficient and cost-effective network hardening using attack graphs. In *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*, pages 1–12. IEEE, 2012.

2. Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM, 2002.
3. Algirdas Avizienis and Liming Chen. On the implementation of n-version programming for software fault tolerance during execution. In *Proc. IEEE COMPSAC*, volume 77, pages 149–155, 1977.
4. H Md Azamathulla, Fu-Chun Wu, Aminuddin Ab Ghani, Sandeep M Narulkar, Nor Azazi Zakaria, and Chun Kiat Chang. Comparison between genetic algorithm and linear programming approach for real time operation. *Journal of Hydro-environment Research*, 2(3):172–181, 2008.
5. Kapil Bakshi. Cisco cloud computing-data center strategy, architecture, and solutions. *CISCO White Paper. Retrieved October*, 13:2010, 2009.
6. Daniel Borbor, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Diversifying network services under cost constraints for better resilience against unknown attacks. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 295–312. Springer, 2016.
7. Benjamin Cox, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser. N-variant systems: a secretless framework for security through diversity. In *Usenix Security*, volume 6, pages 105–120, 2006.
8. Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2):311–338, 2000.
9. Rinku Dewri, Nayot Poolsappasit, Indrajit Ray, and Darrell Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 204–213. ACM, 2007.
10. Rinku Dewri, Indrajit Ray, Nayot Poolsappasit, and Darrell Whitley. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. *International Journal of Information Security*, 11(3):167–188, 2012.
11. Wayne W Eckerson. Three tier client/server architectures: achieving scalability, performance, and efficiency in client/server applications. *Open Information Systems*, 3(20):46–50, 1995.
12. Tom Fifield, Diane Fleming, Anne Gentle, Lorin Hochstein, Jonathan Proulx, Everett Toews, and Joe Topjian. *OpenStack Operations Guide*. " O'Reilly Media, Inc.", 2014.
13. Debin Gao, Michael K Reiter, and Dawn Song. Behavioral distance measurement using hidden markov models. In *Recent Advances in Intrusion Detection*, pages 19–40. Springer, 2006.
14. Miguel Garcia, Alysson Bessani, Ilir Gashi, Nuno Neves, and Rafael Obelheiro. Os diversity for intrusion tolerance: Myth or reality? In *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 383–394. IEEE, 2011.
15. Mukul Gupta, Jackie Rees, Alok Chaturvedi, and Jie Chi. Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach. *Decision Support Systems*, 41(3):592–603, 2006.
16. S. Jajodia, S. Noel, and B. O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challenges*. Kluwer Academic Publisher, 2003.
17. B. Krebs. How many zero-days hit you today? http://krebsonsecurity.com/2013/12/how-many-zero-days-hit-you-today/, 2013.
18. John McHugh. Quality of protection: measuring the unmeasurable? In *Proceedings of the 2nd ACM workshop on Quality of protection*, pages 1–2. ACM, 2006.
19. Peter Mell, Karen Scarfone, and Sasha Romanosky. Common vulnerability scoring system. *Security & Privacy, IEEE*, 4(6):85–89, 2006.

20. Lars Mieritz and Bill Kirwin. Defining gartner total cost of ownership. *L. Mieritz, B. Kirwin*, 2005.
21. Apache mina project. https://mina.apache.org/mina-project/, Oct, 2016.
22. Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.
23. Indrajit Ray and Nayot Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *ESORICS 2005*, pages 231–246. Springer, 2005.
24. Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.
25. Lingyu Wang, Massimiliano Albanese, and Sushil Jajodia. *Network Hardening: An Automated Approach to Improving Network Security*. Springer Publishing Company, Incorporated, 2014.
26. Lingyu Wang, Sushil Jajodia, Anoop Singhal, Pengsu Cheng, and Steven Noel. k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. *Dependable and Secure Computing, IEEE Transactions on*, 11(1):30–44, 2014.
27. Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Steven Noel. k-zero day safety: Measuring the security risk of networks against unknown attacks. In *ESORICS 2010*, pages 573–587. Springer, 2010.
28. Lingyu Wang, Steven Noel, and Sushil Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 2006.
29. Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring the overall security of network configurations using attack graphs. In *Data and Applications Security XXI*, pages 98–112. Springer, 2007.
30. Lingyu Wang, Mengyuan Zhang, Sushil Jajodia, Anoop Singhal, and Massimiliano Albanese. Modeling network diversity for evaluating the robustness of networks against zero-day attacks. In *ESORICS 2014*, pages 494–511. Springer, 2014.
31. Shuzhen Wang, Zonghua Zhang, and Youki Kadobayashi. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers & security*, 32:158–169, 2013.
32. Beytullah Yigit, Gurkan Gur, and Fatih Alagoz. Cost-aware network hardening with limited budget using compact attack graphs. In *Military Communications Conference (MILCOM), 2014 IEEE*, pages 152–157. IEEE, 2014.
33. M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese. Network diversity: A security metric for evaluating the resilience of networks against zero-day attacks. *IEEE Transactions on Information Forensics and Security (TIFS)*, 11(5):1071–1086, 2016.
34. Mengyuan Zhang, Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Massimiliano Albanese. Network diversity: a security metric for evaluating the resilience of networks against zero-day attacks. *IEEE Transactions on Information Forensics and Security*, 11(5):1071–1086, 2016.