Optimizing the Network Diversity to Improve the Resilience of Networks Against Unknown Attacks

Daniel Borbo [a] Lingyu Wang [a] Sushil Jajodia [b] Anoop Singhal [c]

[a] *Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada*

[b] *Center for Secure Information Systems, George Mason University, Fairfax, Virginia, USA*

[c] *Computer Security Division, National Institute of Standards and Technology, Gaithersburg, Maryland, USA*

Abstract

Diversity as a security mechanism is receiving renewed interest due to its potential for improving the resilience of software and networks against previously unknown attacks. Recent works show diversity can be modeled and quantified as a security metric at the network level. However, such efforts do not directly provide a solution for improving the network diversity. On the other hand, existing network hardening approaches largely focus on handling vulnerabilities and do not pay special attention to diversity. In this paper, we propose an automated approach to diversifying network services under various cost constraints in order to improve the network's resilience against unknown attacks. Specifically, we first define models for network services and their relationships, diversification options, and the costs. We then formulate the optimization problem of diversifying network services under given cost constraints. We devise optimization and heuristic algorithms for efficiently solving the problem, and we evaluate our approach through simulations.

Key words: Diversity, network security, optimization, zero day attack

1 Introduction

Today's computer networks are increasingly important to the normal operation of many important infrastructures, such as food supply, electricity grids, transporta-

Email addresses: d_borbor@ciise.concordia.ca (Daniel Borbo), wang@ciise.concordia.ca (Lingyu Wang), jajodia@gmu.edu (Sushil Jajodia).

tion, or communications and public safety systems. Protecting such mission critical computer networks demands more than just preventing known attacks; it is equally important to improve the resilience of such networks against the so-called zero-day attacks exploiting previously unknown vulnerabilities. Although the deployment of traditional defense mechanisms (i.e., firewalls, vulnerability scanners, IDSs, IPSs, etc.) is instrumental in dealing with known attacks, such solutions usually depend on prior knowledge about the attacks and may thus become less effective in dealing with unknown attacks.

For that purpose, diversity has long been considered as a security mechanism for hardening software systems against unknown vulnerabilities, e.g., running different versions of the same software in parallel since an attacker is less likely to compromise all of them [1] (a detailed review of related work will be given in Section 2). While most existing works focus on diversity inside a single software system, some more recent works show that diversity can also be modeled as a network-level security metric, namely, *network diversity* [2,3]. Such works provide a way for formally reasoning about the amount of network diversity in terms of its impact on security, although they do not directly provide a solution for improving the network diversity. On the other hand, there exist many efforts on automatically improving the security of networks, namely, *network hardening*. Nonetheless, those existing network hardening solutions largely focus on dealing with vulnerabilities and do not pay special attention to diversity [4–6]. In contrast to those existing works, our work focuses on optimizing the network diversity in the sense of maximizing its positive impact on the resilience of networks against unknown attacks.

In this paper, we propose an automated approach to diversifying network services under various cost constraints to improve a network's resilience against unknown attacks. Specifically, we devise an extended resource graph model to capture network services and their different instances. Such a model allows us to formulate the diversification requirements and related cost constraints together in an optimization problem. We apply standard optimization techniques on top of heuristic algorithms to efficiently solve the formulated problems for relatively large networks. To evaluate and compare the three different diversity metrics [2,3], we apply them to our hardening problems and analyze their affect on the hardening results. We also evaluate our approach through simulations 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 contributions of this paper are as follows:

- To the best of our knowledge, this is the first automated solution for applying formal network diversity metrics to improve the resilience of networks against unknown attacks [2,3].
- We propose formal models for both network service diversification and its cost, and we also discuss practical aspects of the diversification cost based on Gartner's TCO model [7] and on Emerson-Ponemon Institute's cost of downtime analysis [8].

- As evidenced by the simulation results, our optimization and heuristic algorithms provide a relatively accurate and efficient solution for diversifying network services while respecting various cost constraints.
- By focusing on optimizing diversity, our work provides a complementary solution to existing network hardening approaches which typically focus on dealing with vulnerabilities.

The preliminary version of this paper has previously appeared in [9]. In this paper, we have substantially improved and extended the previous version. The most significant extensions are the following. First, in addition to the shortest path-based metric used in our preliminary version, we now apply all the three network diversity metrics defined in [2,3] in order to evaluate and compare their different effect on the hardening results (Section 3.1, Section 4, and Section 5). Second, we provide an improved cost model and discuss practical aspects of such a model (Section 3.3). Third, we employ new use cases to demonstrate the effect of different network diversity metrics during hardening (Section 4.2). Fourth, we perform a series of new simulations to evaluate and compare the additional network diversity metrics (Section 5). Finally, we have introduced new examples and (e.g., the motivating example in Section 1.1) improved the discussions throughout the paper.

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

1.1 Motivating Example

We first consider a concrete example to demonstrate why diversifying network services for a network infrastructure can be a tedious and error-prone task if done manually, and why it would demand a systematic and automated approach, even if the considered network is of a relatively small size. Figure 1 shows a hypothetical network roughly based on Cisco's cloud data center concept [10] as well as the OpenStack architecture [11]. Despite its relatively small scale, it mimics a typical cloud network, e.g., the network consists of different layers for web, application, and database or storage services. The client layer connects the cloud network to the internet through the CRS 7600; a firewall (ASA v1000) separates the external network from the internal one. There is a security/authentication layer (authentication server, Neutron server, etc.) in addition to the 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 [12].

We make the following assumptions about the network. We assume four different

types of users may connect to this network [13,12]: *i*) a normal cloud user (*NU*), *ii*) a cloud tenant (*CT*), *iii*) a cloud provider (*CP*) and *iv*) a third party cloud provider (*CP3P*). We assume the database servers are the most critical assets in the cloud and would focus on potential attacks directed toward such servers. We use Amazon's multi-tier infrastructure concept [14] to inter-connect different VM servers. Following this model, for example, the normal user (*NU*) would first connect to a web server, then to an application server, before finally connecting to a database server. We assume that all VMs providing the same service will run on the same physical server cluster (e.g. all $http$ VMs run on the $http$ server cluster). Additionally, we assume the network is secured against known vulnerabilities and we will not consider exploits and conditions that involve the firewalls. Finally, all physical machines and VMs run ssh for maintenance. Figure 2 shows an attack graph for four users who have different initial privileges (the tuple inside each oval represents the exploit of a vulnerability from a source host to a destination, and each pair shown in clear text represents a pre- or post-condition of such exploits).

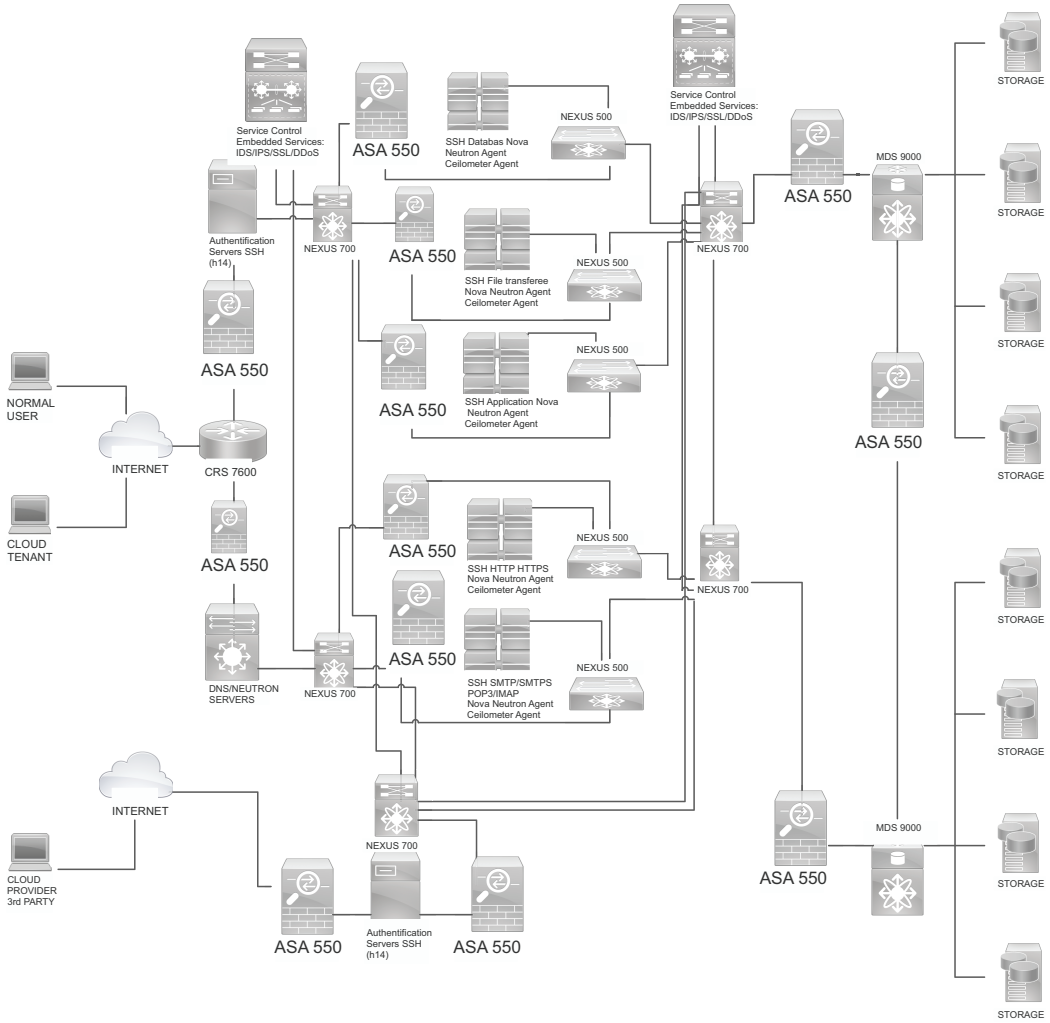


Fig. 1. The example network.

To measure the network's resilience against unknown zero-day attacks, we con-

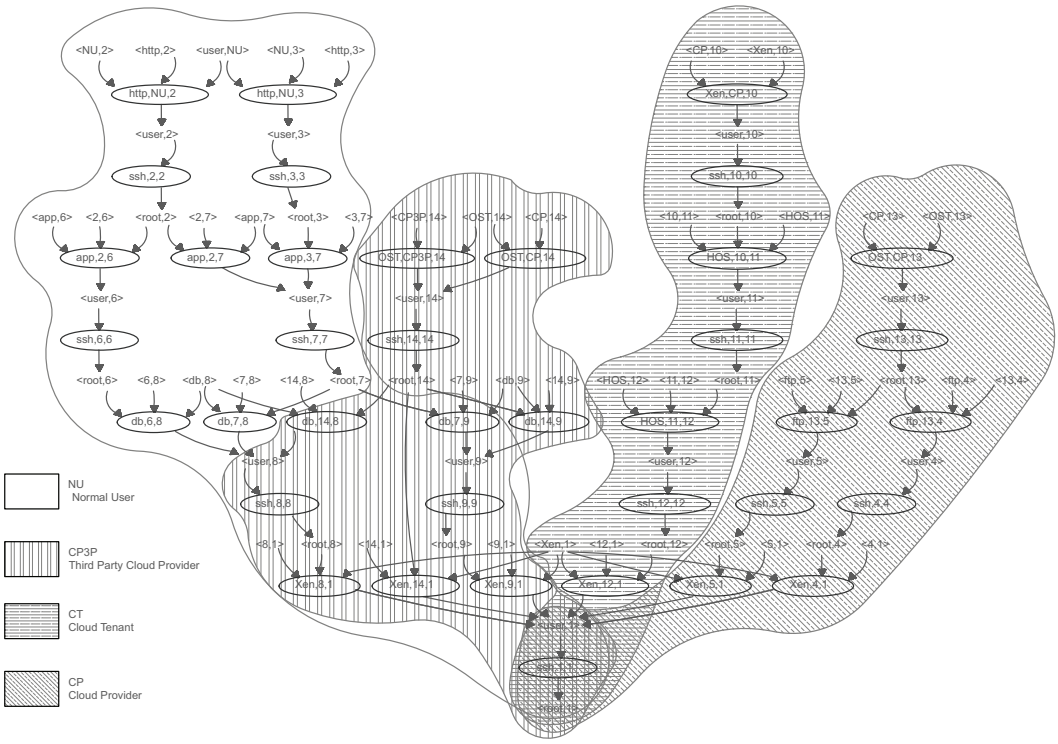


Fig. 2. The example network's attack graph.

sider the *network diversity metrics* [2,3]. While we will apply all the three network diversity metrics to cover different aspects of the diversity property, we will focus on the d_2 metric in this example for simplicity. The d_2 metric counts the number of distinct resources inside a network, while considering the uneven distribution of resources and varying degree of similarity between resources [3]. For simplicity, although the attacker may follow many paths to compromise the storage servers, here we only consider one of the web servers as the initial targets. We can observe that the attacker would need to exploit at least five distinct zero-day vulnerabilities, one for the web server, one for the application server it connects to, one for the database server, one for the Xen server, and one to escalate privileges on each one of these servers [1] . Finally, we assume the administrator has the option of replacing the Web servers running IIS 10.0 with either Apache 2.4.23, NGINX 1.9, Litespeed 5.0.14, Cherokee 1.2.104 or GWS 2.1; the database server running Oracle 10.gR2 can be replaced with either MySQL 5.7.16, MSSQL 2014, Mongo 3.2, PostgreSQL 9.1 or DB2 11.1; the OpenSSH 5.0.10 can be replaced with either Apache MINA 1.0, GeorgiaSoftWorks 8.07, Copssh 5.5.3, Dropbear 2016 or Pragma 5.0. We also assume there are five different instances for the application service and no change can be made to OpenStack. Finally, we assume any of these changes will incur a given installation/maintenance cost (we will discuss the cost model in more details later in Section 3). Based on above assumptions, we may consider potential network hardening scenarios as follows.

[1] If different software is considered likely to share common vulnerabilities, a similarity-sensitive diversity metric may be needed [2,3].

- *Scenario 1:* The administrator aims to optimally diversify the network (i.e., maximizing the aforementioned d_2 metric) in order to render the network as resilient to zero-day attacks as possible.
- *Scenario 2:* The same goal as in above Scenario 1, but under the constraint that the overall diversification cost must be less than a given budget.
- *Scenario 3:* The same goal as in above Scenario 2, with an additional constraint that the SSH services cannot be replaced.
- *Scenario 4:* The same goal as in above Scenario 3, with an additional constraint that replacing the IIS web server ($h3$) should be given a higher priority.
- *Scenario 5:* The same goal as the above while considering multiple diversity metrics at the same time.

Clearly, finding the optimal hardening solution in those scenarios is not always straightforward even for such a toy example. Considering that the attacker may come under different user accounts armed with different initial privileges (e.g., as a cloud user, tenant, provider, or third-party provider), the problem becomes even more complicated. This shows the need for an automated approach, which will be the subject matter of this paper.

2 Related Work

In general, the security of networks may be qualitatively modeled using attack trees [5,15,16] or attack graphs [17,18]. A majority of existing quantitative models of network security focus on known attacks [19,20], while few works have tackled zero-day attacks [21,22,2,3] which are usually considered unmeasurable due to the uncertainties involved [23]. Early works on network hardening typically rely on qualitative models while improving the security of a network [18,4,24]. 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.

One of the first attempts to provide a systematic cost model to deal with budget constraints is by Gupta et al. [25]. The authors employed genetic algorithms to solve the problem of choosing the best set of security hardening options while reducing costs. Dewri et a. [5] build on top of Gupta's work to address the network hardening problem using a more systematic approach which consider both a single objective optimization problem and the multiple objective variations. Their work considers the damage of compromising any node in the cost model to determine the most cost-effective hardening solution. Later, in [15] and in [26], the authors extrapolate the network hardening optimization problem as vulnerability analysis with cost/benefit assessment and risk assessment, respectively. In [27], Poolsapp-

asit et al. extend Dewri's model to also consider dynamic conditions (conditions that may change or emerge while the model is running) by using Bayesian attack graphs to consider the likelihood of an attack. Unlike our work which focuses on optimizing the diversity, most of those existing work on network hardening focus on handling known vulnerabilities through disabling existing services.

There exist many research works on extending attack trees and attack graphs to security metrics, as surveyed in [28]. A probabilistic metric is applied to attack graphs to obtain an overall attack likelihood for the network [30]. A Bayesian Network (BN)-based security metric applies attack graphs to measure the security level of a network [31]. The metric converts the CVSS scores of vulnerabilities into attack probabilities and then obtains the overall attack likelihood for reaching critical assets. The National Institute of Standards and Technology (NIST) highlights the importance of using a security metrics on cloud systems and provides frameworks and definitions for this purpose [32]. Most of the works assign numeric scores to rank known vulnerabilities (mostly based on the CVSS) [29] in order to model the severity or impact that they may have on a network. This ranking is usually based on how likely and easily exploitable the known vulnerabilities are. This ranking, however, is not always possible for unknown vulnerabilities which lack the information required for such a ranking. The k-zero-day safety metric [21,22] is the first to address this limitation, which counts at least how many zero-day vulnerabilities are needed to compromise a critical asset without ranking them.

There exists 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 [1], and similar ideas have been employed for preventing security attacks, such as the N-Variant system [33] and the behavioral distance approach [34]. In addition to design diversity and generated diversity, recent work employs opportunistic diversity which already exists among different software systems. For example, the practicality of employing OS diversity for intrusion tolerance is evaluated in [35]. More recently, the authors in [2,3] adapt biodiversity metrics to networks and lift the diversity metrics to the network level. While those works on diversity provide a foundation to our work, they do not directly provide a systematic solution for improving diversity. Finally, our discussions are mostly based on a cloud network similar to what is introduced in [12] which applies different threat models to fictitious but realistic cloud infrastructures.

3 Model

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

3.1 Extended Resource Graph

The first challenge is to model different resources, such as services (e.g., Web servers) that can be remotely accessed over the network, different instances of each resource (e.g., Apache and IIS), and the causal relationships existing among resources (e.g., a host is only reachable after an attacker gains a privilege to another connected host). For this purpose, we will extend the concept of *resource graph* introduced in [2,3], which is syntactically equivalent to attack graphs, but models network resources instead of known vulnerabilities as in the latter. Specifically, we propose the extended resource graph to introduce the notion of *Service Instance* to indicate which instance (e.g., Apache) of a service (e.g., Web server) is being used on a host. Like the original resource graph, we only consider services that can be remotely accessed.

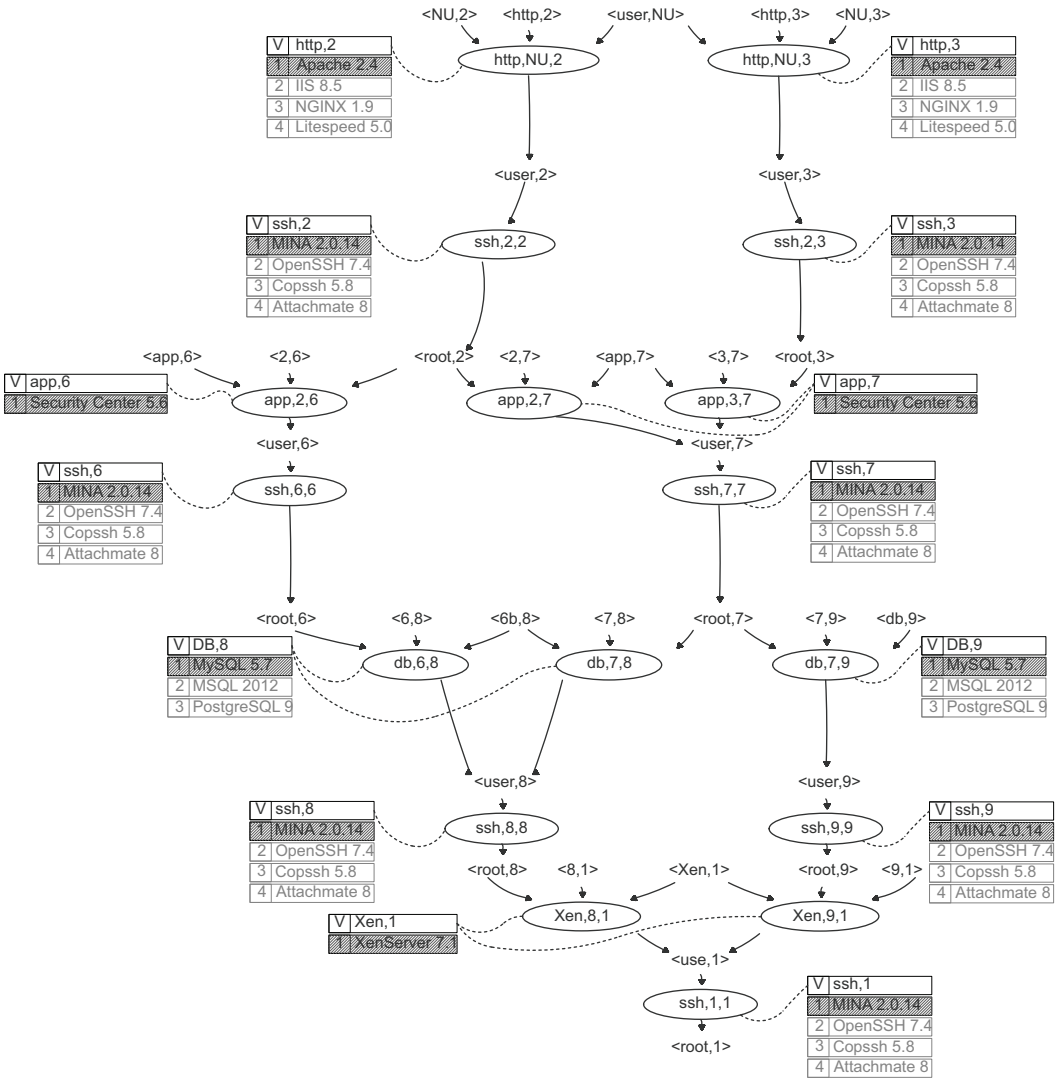


Fig. 3. The extended resource graph of our running example.

The extended resource graph of the running example is shown in Figure 3. Similar to the attack graph discussed in the previous section, in Figure 3, each pair shown in plaintext is a security-related condition (e.g., connectivity $\langle source, destination \rangle$ or privilege $\langle privilege, host \rangle$). Each exploit node (shown as an 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). The small one-column table beside each exploit indicates the current service instance using a highlighted index (e.g., 1 means Apache and 2 means IIS) and other potential instances in lighter text. The edges point from pre-conditions to an exploit (e.g., from $\langle 1, 2 \rangle$ and $\langle http, 2 \rangle$ to $\langle http, 1, 2 \rangle$), and from the exploit to its post-conditions (e.g., from $\langle http, 1, 2 \rangle$ to $\langle user, 2 \rangle$).

A design choice here is whether to associate the service instance concept with a condition indicating the service (e.g., $\langle http, 2 \rangle$) or the corresponding exploits (e.g., $\langle http, 1, 2 \rangle$). While it is more straightforward to have the service instance defined as the property of a condition, which can then be inherited by the corresponding exploits, we have opted to define this property as a label for the exploit nodes in the graph, because this will make it easier to calculate the number of distinct services along a path which is critical to our metrics, as we will explain later. One complication then is that we must ensure all exploits with the same service and destination host (e.g., $\langle http, 1, 2 \rangle$ and $\langle http, 3, 2 \rangle$) to be associated with the same service instance. Definitions 1 and 2 formally introduce these concepts.

Definition 1 (Service Pool and Service Instance) *Denote S the set of all services and Z the set of integers, for each service $s \in S$, the function $sp(.) : S \rightarrow \mathsf{Z}$ gives the service pool of s which represents all available instances of that service.*

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

- *a set of hosts H,*
- *a set of services S, with the service mapping $serv(.) : H \rightarrow 2^S$,*
- *the collection of service pools $SP = \{sp(s) \mid s \in S\}$,*
- *and the labelling function $v(.) : E \rightarrow SP$, which satisfies $\forall h_s \in S \ \forall h'_s \in S, v(\langle s, h_s, h_d \rangle) = v(\langle s, h'_s, h_d \rangle)$ (meaning all exploits with common service and destination host must be associated with the same service instance, as explained earlier).*

Let E be the set of zero-day exploits $\{\langle s, h_s, h_d \rangle \mid h_s \in H, h_d \in H, s \in serv(h_d)\}$, C be the set of pre- and post-conditions of the exploits in E, 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.

3.2 Diversity control

We employ the notion of *diversity control* as a model for diversifying one or more services in the resource graph. Since we represent the service instance using integers, it will be straightforward to regard each pair of service and destination host on which the service is running as an optimization variable, and formulate diversity 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 conditions indicating services, instead of the number of exploits (since many exploits may share the same service instance, and hence the optimization variable). 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 network (i.e., the number of hosts). We will further evaluate and discuss the scalability of our solution in Section 5.

Definition 3 (Optimization Variable and Diversity Control) *Given an extended resource graph $\langle G, v\rangle$, $\forall e \in E,\ v(e)$ is an optimization variable. A diversity control vector is the integer valued vector $\vec{V} = (v(e_1), v(e_2), ..., v(e_{|E|})$.*

Changing the value of an optimization variable has an associated *diversification cost* and the collection of such costs is given in a *diversity cost matrix* in a self-explanatory manner. Like in most existing works (e.g., [5,27,15]), we assume an administrator can estimate the diversification costs based on monetary, temporal, and scalability criteria like *i*) installation cost, *ii*) operation cost, *iii*) training cost, *iv*) system downtime cost and, *v*) incompatibility cost. Taking this criteria as a point of reference, subsection 3.3 provides a guideline on how to instantiate our cost model and how to calculate these diversification costs.

The following defines the diversity cost, diversity cost matrix, and the total diversity cost.

Definition 4 (Diversification Cost and Diversity Cost Matrix) *Given $s \in S$ and $sp(s)$, the cost to diversify a service by changing its service instance inside the service pool is called the diversification cost. The collection of all the costs constraints associated with changing services in S are given as a diversity cost matrix DCM in which the element at i^{th} row and j^{th} column indicates the diversification cost of changing the i^{th} service instance to be the j^{th} service instance. Let $v_s(e_i)$ be the service associated with the optimization variable $v(e_i)$ and $\vec{V}_0$ the initial service instance values for each of the exploits in the network. The total diversification cost, Q_d, given by the diversity vector $\vec{V}$ is obtained by*

$$Q_d = \sum_{i=1}^{|E|} DCM_{v_s(e_i)}(\vec{V}_0(i), \vec{V}(i))$$

We note that the above definition of diversification cost between each pair of service instances has some advantages in practice. For example, we can easily imagine cases in which 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.

Moreover, our cost model can be used to specify many different types of cost constraints which can be added to the base formula. For example, an administrator might have configured service groups to group related services together (e.g., SIP, RTP, and RTSP) and a change in one service might also affect the others. In other words, the way our costs are calculated allow them to be derived as a function of the status of other services or conditions.

Another important advantage of our model is the inclusion of negative costs. While at a first glance this concept may not seem self-evident, the inclusion of negative cost values can be interpreted as an incentive to opt for a specific option. For example, an administrator may want to phase out the use of rsh in favor of a more secure protocol like ssh. This can be easily represented by negative cost values within our two-dimensional matrix which effectively subtracts costs from the total hardening cost.

3.3 Cost Estimation

Although how administrators may choose to instantiate their cost models will eventually depend on their specific applications and needs, the general methodology for deriving a baseline cost will still be valuable. Therefore, we make use of Gartner's 2003's *Total Cost of Ownership* (TCO) analysis report [7] and Emerson-Ponemon Institute's 2016's analysis report on the cost of data center outages [8] to discuss how a realistic estimation of real world costs may be obtained for diversifying one or more services.

Based on Gartner's report, a company's costs can be divided into two main categories: Base costs and ongoing costs. The base costs are mostly associated with planning costs that include, but are not limited to, server/software acquisition and installation costs. The ongoing costs are the costs of having a server or a service up and running. The ongoing costs are further divided into direct and indirect costs which include operational costs and downtime costs, respectively. A more detailed list can be seen in Table 1. It can be observed that direct costs (like security management costs, changes in upgrade costs or productions costs), as well as indirect costs (like downtime cost), are costs that need to be considered before diversifying a service. An administrator may apply such a TCO model to his/her specific scenario to calculate the total costs for diversification.

In practice, it might also be infeasible for an administrator to apply the entire TCO model to calculate the diversification costs. Instead, he/she may decide to estimate the cost using one or more aspects of the model, e.g., the ongoing costs, as a reference point. Additionally, it is shown that the ongoing costs alone will incur, on average, around 85% of the total costs of ownership, and therefore using only the ongoing costs to estimate diversification costs could give a reasonable estimation of the total cost.

For example, we consider one particular indirect ongoing cost, namely, the system downtime cost. In Emerson-Ponemon's 2016's [8] report on the downtime costs of a data center, the impact that downtime costs can have on a network is highlighted. Such industry benchmarks and insights would allow a system administrator to make the right business decisions to minimize costly downtimes while hardening the network. An estimation of the system downtime is provided in [36] as follows:

$$\bar{q}_{hr(dt)} = \bar{E}_{q(hr)} \times \breve{E}_{af} + \bar{R}_{hr} \times \breve{R}_{af}$$

Where

- $\bar{q}_{hr(dt)}$ is the estimated average cost of one hour of downtime,
- $\bar{E}_{q(hr)}$ is the estimated average employee costs per hour. It is the total salaries and benefits of employees per week divided by the average number of working hours. It is the total revenue per week divided by average number of open hours.
- $\breve{E}_{af}$ is the estimated fraction of employees affected by the downtime,
- $\bar{R}_{hr}$ is the estimated average revenue per hour, and
- $\breve{R}_{af}$ is the estimated fraction revenue affected by the downtime.

Depending on the exact methodology for diversifying services, the different downtime components may have a significant (e.g., replacing a popular desktop software may affect a large fraction of employees, and hence lead to a large $\bar{E}_{q(hr)} \times \breve{E}_{af}$ value) or negligible (e.g., replacing a server-side service through live migration does not result in much revenue loss, and hence $\bar{R}_{hr} \times \breve{R}_{af}$ could be very small) value. Some of those measures, such as the *Fraction Employees Affected by Outage* and the *Fraction Revenue Affected by Outage*, might not be readily available and need to be based on an educated guess on plausible ranges.

As an example to better illustrate this, we discuss the reported 2015 revenue for Amazon. This revenue was reported at approximately \$107 billion [37] with approximately 250,000 employees for that same year [38]. From this information, the approximate revenue per hour (considering that Amazon is a 24/7 business) is about \$12M. Assuming an average annual salary of an employee being around \$100,000 then we can have approximate yearly expenditure of \$25B on salaries or approximately \$471M per week for all staff. If we consider that an Amazon employee works on average 50 hours per week, then the average expenditure per salary per hour is around \$9.4M per hour. We assume that if an outage for the

Gartner's TCO's base costs		
Planning costs (Approx. 15% of TCO)	Acquisition costs	Cost of Hardware
		Cost of OS
		Cost of Application
	Installation costs	Hardware setup
		OS installation
		Application installation
Gartner's TCO's ongoing costs		
Indirect costs (Approx. 50% of TCO)	Downtime costs	Planned downtime
		Unplanned downtime
	End-user costs	Casual learning
		Peer and self support
Direct costs (Approx. 35% of TCO)	Operational costs	Communication fees
		Leased asset fees
		IS commodity expenditures
		Insurance
	Support costs	Help desk
		Request and problem management
		Casual learning
		Training
		Operating costs
	Changes in upgrade costs	Change planning
		Asset management
		Product evaluation and testing
		Product procurement and implementation
		User administration
	Security management and failure control costs	Security and virus protection
		LAN/WAN troubleshooting/repair
		Disaster planning and recovery
		Hardware maintenance fees
	Monitoring costs	Event management
		Performance management
		Physical site management
	Production control costs	Application management
		Storage management
		Traffic management

Table 1
Gartner's TCO Costs

ftp services affects 84% of the revenue, that would equate to a loss of around \$10M. If it affects 85% of the employees, then that would equate to approximately \$8M. Thus, the total revenue loss for an outage would be valued at approximately $\bar{q}_{hr(dt)} = \$9.4M \times 0.85 + \$12M \times 0.84 = \$18M$. This value can be used as a base monetary reference to define the costs to diversify the ftp service.

Finally, while this kind of estimation provides a good starting point toward a realistic diversity cost for network administrators and security consultants, it can certainly be refined, e.g., by considering outage prevention mechanisms.

3.4 Diversity Metrics

We will apply the network diversity metrics originally proposed in [3] to measure the level of resilience against unknown attacks. We demonstrate the effect of diversification in terms of each of the three d-diversity metrics through two examples. Example 1 (before diversification) shows an extended resource graph where none of the services have been diversified; on the other hand, Example 2 (after diversification) shows an extended resource graph where all the services have been diversified.

First, the d_1 metric basically counts the number of distinct resources in a network while considering the uneven distribution of resources. This metric is mostly used to evaluate the scale of potential infection by a malware, and it is also a building block of the other two metrics.

Definition 5 (Effective Richness and d_1-Diversity [3]) *In an extended resource graph* $\langle G(E \cup C, R_r \cup R_i), v\rangle$ *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). We define the network's diversity a* $d_1 = \frac{r(G)}{t}$ *where* $r(G)$ *is the network's effective richness of the services, defined as* $r(G) =$

$$\frac{1}{\prod_1^n p_i^{p_i}}$$

In Figure 4, Example 1 shows that all three $http$ services share the same service instance. The same thing can be said for all four ssh services. For Example 2 in the same figure, there is only one instance per unique service. From this example, it is clear that d_1 metric provides a general measure for the level of diversity among services and for the resilience of networks against zero-day attacks, although it does not consider any relationships between the services.

Unlike the d_1 metric, the d_2 metric more specifically measures diversity against a given critical asset based on the least attacking effort required to compromise said asset. Partially based on the k-zero-day safety metric [22], this second metric is more suitable to measure a network's capability to resist zero-day attacks that require multiple related attacks in an attack path.

For the d_2 metric, we describe the concept of attack path in Definition 6 and how they are used within the extended resource graph. By analyzing the different attack

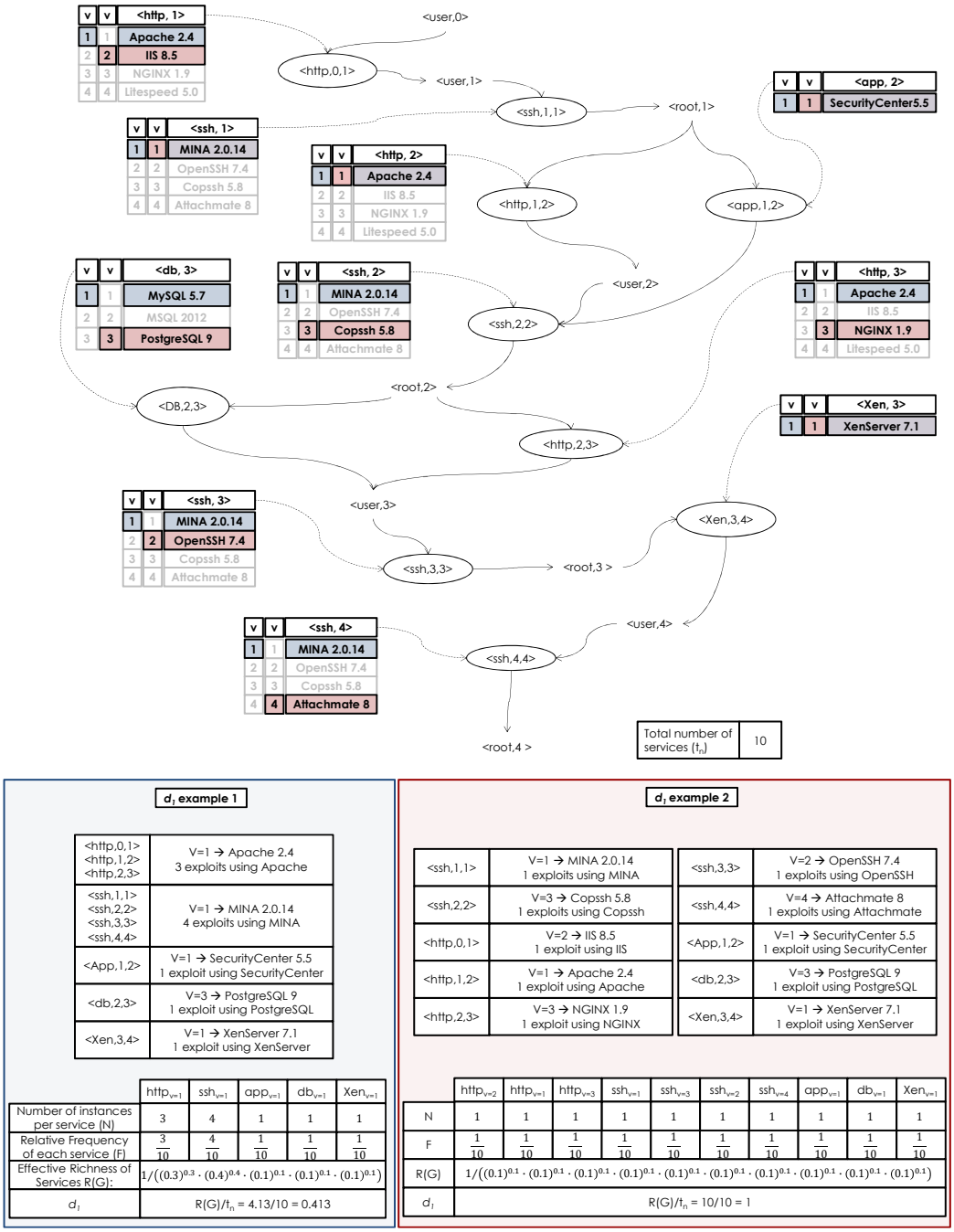


Fig. 4. d_1 metric example.

paths in the extended resource graph, Example 1 in Figure 5 shows that the minimum number of distinct services is three and the minimum number of steps when considering all attack paths is seven. In contrast, Example 2 on the same Figure shows these values to be both seven. By using Definition 7, we can see that the d_2 metric increases when services are diversified. The greater the value of the d_2 metric, the better the resilience against zero-day attacks with respect to the given critical asset.

Definition 6 (Attack Path [3]) *Given an extended resource graph* $\langle G(E \cup C, R_r \cup R_i), v \rangle$*, we call* $C_I = c : c \in C, (\nexists e \in E)((e, c) \in R_i)$ *the set of initial conditions. Any sequence of zero day exploits* $e_1, e_2, \cdots, e_n$ *is called an attack path in G, if* $(\forall i \in [1, n])((c, e_i) \in R_r \rightarrow (c \in C_i \vee (\exists j \in [1, i1])((e_j, c) \in R_i)))$*, and for any* $c \in C$*, we use* $seq(c)$ *for the set of attack paths* $e_1, e_2, \cdots, e_n : (e_n, c) \in R_i$*.*

Definition 7 (d_2**-Diversity [3])** *Given an extended resource graph* $\langle G(E \cup C, R_r \cup R_i), v \rangle$ *and a goal condition* $c_g \in C$*, for each* $c \in C$ *and* $q \in seq(c)$*, denote* $R(q) for s : s \in R, rappear sinq$*, the network diversity is defined as (where min(.) returns the minimum value in a set)* $d_2 = min_{q \in seq(c_g)} d_1(R(q))$

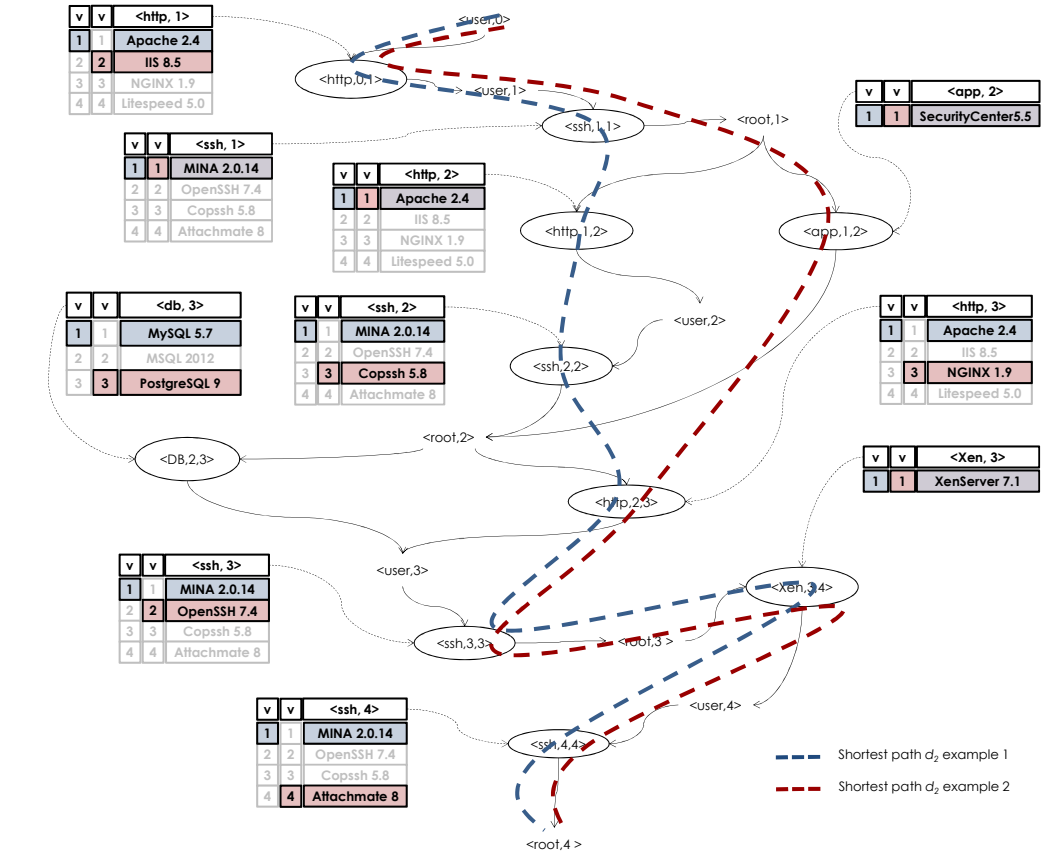
In contrast to the d_2 metric, which models the least attacking effort, the d_3 metric is a conditional probability that depicts the average attack effort (likelihood) required by an attacker to compromise a critical asset. This Bayesian network-based metric is intended as a complementary metric to the d_2 metric in measuring diversity with respect to a given critical asset.

Definition 8 (d_3 **Diversity)** *Given an extended resource graph* $\langle G(E \cup C, R_r \cup R_i), v \rangle$*,*

(1) for each $e \in E$*, a given conditional probability* $P(e \mid \bigwedge_{c:(c,e) \in R_r} c = TRUE)$*,*
(2) conditional probabilities $P(e \mid \bigwedge_{c:(c,e) \in R_r} c = FALSE) = 0$*,*
(3) conditional probabilities $P(c \mid e = TRUE \wedge (e, c) \in R_i) = 1$*, and*
(4) for any $e_1, e_2 \in E$ *involving the same service s, conditional probabilities* $P(e_1 \mid e_2 = TRUE \wedge (\bigwedge_{c:(c,e_1) in R_r} c) = TRUE)$ *and* $P(e_2 \mid e_1 = TRUE \wedge and_{\, c:(c,e_2) \in R_r} c = TRUE))$*,*

Given any $c_g \in C$*, the network diversity* d_3 *is defined as* $d_3 = \frac{p'}{p}$ *where* p *denotes the conditional probability of* c_g *being satisfied given that all the initial conditions are true, and* p' *denotes the probability of* c_g *being satisfied given that all initial conditions are true and the above fourth set of probabilities are not given (i.e., without considering the effect of reusing any exploit).*

Figure 6 presents how diversifying services can affect the probability of compromising the critical asset ($< root, 4 >$). In the resource graph, we have added two additional nodes to express the probability that an attacker can compromise a service with multiple instances (named <*Can do* $http_{v=1}$>*, and* <*Can do* $ssh_{v=1}$>. Next to each of the exploits, we have added the initial probabilities that the attacker can exploit. Below the attack graph, we have added the marginal probability distribution of some nodes to show how diversifying services can affect the d_3 metric. We can see that, as we diversify the network further, the probability to compromise the critical asset decreases.



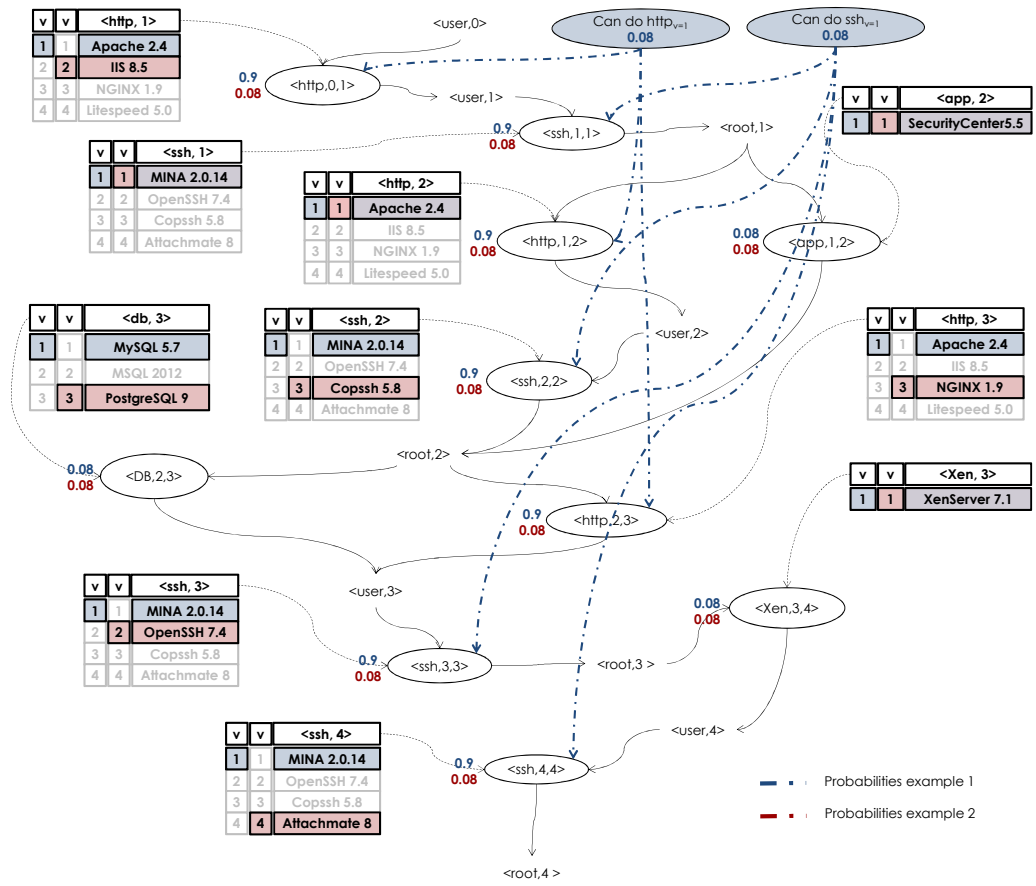
d_2 example 1	Number distinct Services	Number of steps on path	d_2
<http$_{v=1}$,0,1> → <ssh$_{v=1}$,1,1> → <http$_{v=1}$,1,2> → <ssh$_{v=1}$,2,2> → <http$_{v=1}$,2,3> → <ssh$_{v=1}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=1}$4,4>	3	8	3/7 Or 0.4286
<http$_{v=1}$,0,1> → <ssh$_{v=1}$,1,1> → <http$_{v=1}$,1,2> → <ssh$_{v=1}$,2,2> → <db$_{v=1}$,2,3> → <ssh$_{v=1}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=1}$4,4>	4	8	
<http$_{v=1}$,0,1> → <ssh$_{v=1}$,1,1> → <app$_{v=1}$,1,2> → <http$_{v=1}$,2,3> → <ssh$_{v=1}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=1}$4,4>	4	7	
<http$_{v=1}$,0,1> → <ssh$_{v=1}$,1,1> → <app$_{v=1}$,1,2> → <db$_{v=1}$,2,3> → <ssh$_{v=1}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=1}$4,4>	5	7	

d_2 example 2	Number distinct Services	Number of steps on path	d_2
<http$_{v=2}$,0,1> → <ssh$_{v=1}$,1,1> → <http$_{v=1}$,1,2> → <ssh$_{v=3}$,2,2> → <http$_{v=3}$,2,3> → <ssh$_{v=2}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=4}$4,4>	8	8	7/7 Or 1
<http$_{v=2}$,0,1> → <ssh$_{v=1}$,1,1> → <http$_{v=1}$,1,2> → <ssh$_{v=3}$,2,2> → <db$_{v=1}$,2,3> → <ssh$_{v=2}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=4}$4,4>	8	8	
<http$_{v=2}$,0,1> → <ssh$_{v=1}$,1,1> → <app$_{v=1}$,1,2> → <http$_{v=3}$,2,3> → <ssh$_{v=2}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=4}$4,4>	7	7	
<http$_{v=2}$,0,1> → <ssh$_{v=1}$,1,1> → <app$_{v=1}$,1,2> → <db$_{v=1}$,2,3> → <ssh$_{v=2}$,3,3> → <Xen$_{v=1}$,3,4> → <ssh$_{v=4}$4,4>	7	7	

Fig. 5. d_2 metric example.

3.5 Problem formulation

A closer look at Figure 3 to Figure 6 shows that if we increase the number of distinct resources in a network, its overall resilience is increased. This is reflected for all three d-diversity metrics. To maximize the resilience of a network, the d_1 and d_2



d_3 example 1

<http$_{v=1}$,0,1>	0.072	<ssh$_{v=1}$,1,1>	0.0052
<root,1>	0.0052	<http$_{v=1}$,1,2>	0.0047
<app$_{v=1}$,1,2>	0.0004	<ssh$_{v=1}$,2,2>	0.0042
<root,2>	0.0003	<db$_{v=1}$,2,3>	2.69×10^{-5}
<http$_{v=1}$,2,3>	0.0003	<ssh$_{v=1}$,3,3>	2.18×10^{-5}
<root,3>	2.18x10-5	<Xen$_{v=1}$,3,4>	1.74×10^{-6}
<ssh$_{v=1}$,4,4>	1.57×10^{-6}	<root,4>	1.57×10^{-6}
d_3	1.57×10^{-6}	$Log_{0.08} d_3$	5.29

d_3 example 2

<http$_{v=2}$,0,1>	0.08	<ssh$_{v=1}$,1,1>	0.0064
<root,1>	0.0064	<http$_{v=1}$,1,2>	0.0005
<app$_{v=1}$,1,2>	0.0005	<ssh$_{v=3}$,2,2>	4.096×10^{-5}
<root,2>	3.27×10^{-6}	<db$_{v=1}$,2,3>	2.62×10^{-7}
<http$_{v=3}$,2,3>	2.62×10^{-7}	<ssh$_{v=2}$,3,3>	1.68×10^{-9}
<root,3>	1.68×10^{-9}	<Xen$_{v=1}$,3,4>	1.34×10^{-10}
<ssh$_{v=4}$,4,4>	1.07×10^{-11}	<root,4>	1.07×10^{-11}
d_3	1.07×10^{-11}	$Log_{0.08} d_3$	10

Fig. 6. d_3 metric example.

values also need to be maximized, while the d_3 value needs to be minimized. This can be achieved by changing the service instances if we respect the available budget of cost.

For example, taking Figure 6 as a reference, let's assume that an administrator has an available budget of \$7.8k, and the cost to diversify the ssh service from MINA 2.0.14 (service instance $v = 1$) to OpenSSH 7.4 ($v = 2$), Copssh 5.8 ($v = 3$) or Attachmate 8 ($v = 4$) is \$7.8k, \$1.2k, and \$3.4k respectively. We can see

that changing $\langle ssh, 3, 3\rangle$ from instance $v = 1$ to $v = 2$ would respect the budget and would increase values of the network's d-diversity metrics. We may also see that this is not the optimal solution, since we could also replace $\langle ssh, 2, 2\rangle$ and $\langle ssh, 4, 4\rangle$ with instances $v = 3$ and $v = 4$, respectively, further increasing the d-diversity metrics and still respecting the available budget.

As demonstrated here and in Section 1.1, for any realistic networks, it will not be feasible to manually perform such calculations and obtain the optimal results. Therefore, in the following, we formulate the problem of obtaining the optimal d-diversity as three different optimization problems.

Problem 1 (d_1 Optimization Problem) *Given an extended resource graph $\langle G, v\rangle$, find a diversity control vector $\vec{V}$ which maximizes $min(d_1(\langle G(\vec{V}), v\rangle))$ subject to the constraint $Q \leq B$, where B is the available budget and Q is the total diversification cost as given in Definition 4.*

Problem 2 (d_2 Optimization Problem) *Given an extended resource graph $\langle G, v\rangle$, find a diversity control vector $\vec{V}$ which maximizes $min(d_2(\langle G(\vec{V}), v\rangle))$ subject to the constraint $Q \leq B$, where B is the available budget and Q is the total diversification cost as given in Definition 4.*

Problem 3 (d_3 Optimization Problem) *Given an extended resource graph $\langle G, v\rangle$, find a diversity control vector $\vec{V}$ which minimizes $max(d_3(\langle G(\vec{V}), v\rangle))$ subject to the constraint $Q \leq B$, where B is the available budget and Q is the total diversification cost as given in Definition 4.*

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 [39] has seen many real applications to enterprise networks) may be easily extended to generate the extended resource graphs we need as inputs. Additionally, our problem formulation assumes a very general 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 explained in the following section.

4 Methodology

This section details the optimization and heuristic algorithms used for solving the formulated diversification problems and describes a few use cases.

4.1 Optimization Algorithm

Our first task is to select an optimization algorithm that is suitable for solving the hardening problem. Roughly speaking, there exist mainly two types of optimization algorithms: Conventional methods or exact algorithms and meta-heuristic approaches [40]. Exact (gradient-based) algorithms, such as Lagrangian relaxation and branch and bound, consider all the solution spaces to give a global solution [41]. However, it is well known that most of these methods require to satisfy mathematical properties like convexity or differentiability [42], which may not always be applicable to our problem. The problem we want to solve includes different if-then-else constructs to account for the different cost constraints used, and thus, an algorithm that allows to insert this construct is necessary. Meta-heuristic approaches, such as genetic algorithm, particle swarm optimization, imperialist competitive algorithm, etc., consider some parts of the solution space to reach a global optimum or near-solution optima, which provides an advantage when dealing with discrete variable spaces [41], which closely match the requirement of our hardening problem. They provide a simple and robust search method and optimization technique. Because the problem we want to solve uses variables that are defined as discrete, a meta-heuristic approach is needed.

The genetic algorithm (GA) provides a simple and clever way to encode candidate solutions to the problem [43]. One of the main advantages is that we do not have to worry about explicit mathematical definitions (which allow for a quick implementation). For our automated optimization approach, we chose GA, which is popular among the different evolutionary algorithms due to certain characteristics: It requires little information to search effectively in a large search space in contrast to other optimization methods (e.g., the mixed integer programming [41]); and that it uses both crossover and mutation operators which makes its population more diverse and thus more immune to be trapped in some local optima. While our work was inspired by [5], our main difference and contribution is that we focus on service diversification and not on disabling services.

The extended resource graph is the input to our automated optimization algorithm where the function to be optimized (fitness function) is each of the three d-diversity metrics applied to the extended resource graph. One important point to consider when optimizing the d metric functions on the extended resource graph is that, for each generation of the GA, the graph's service instance labels will dynamically change. This in turn will change the value of d-diversity metrics since the actual configuration of the graph may have changed with each successive generation of the GA. Our optimization tool takes this into consideration. For the d_2 metric we note one limitation. Our optimization tool does not provide a priority if there are more than one shortest path that provide the optimized d_2 since the optimization only aims at maximizing the minimum d_2.

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 diversity cost Q_d) or specific constraints to address certain requirements or priorities while diversifying services (e.g., the cost to diversify $http$ services should be less than 80% of the cost to diversify ssh; if it is strongly encouraged to use OpenSSH 7.4 for the ssh service, it's cost could be represented with a negative cost (credit) and thus incentivizing its selection; etc.). Those constraints are specified using the diversity control matrix.

The number of independent variables used by the GA (genes) are the optimization variables given by the extended resource graph. For our network hardening problem, the GA will be dealing with integer variables representing the selection of a hardening option. Because $v(e)$ (optimization variable) 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. 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 diversity control vector $\vec{V}$ is obtained after running the GA.

4.2 Use Cases

In the following, we demonstrate different use cases of our method with varying cost constraints and hardening options. For these use 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). Specifically, for the use cases discussed below, we have used the following algorithm parameters: population size = 100, number of generations = 150, crossover probability = 0.8, and mutation probability = 0.2.

Use Case A: $Q_d \leq$ \$300K with individual constraints per service. We start with the simple case of one overall budget constraint (refer to Figure 7). There are 14 different services-based optimization variables. The solution provided by the GA for each one of the diversity metrics is $d_1 = 0.555458$, $d_2 = 3.464102$, and $d_3 = 3.12 \times 10^{-7}$. The total incurred cost was \$276K which is within the budget Q_d ($Q_d \leq$ \$300K).

On the other hand, if we assign individual budgets per services, while maintaining the overall budget $Q_d \leq$ \$300K, the optimization results will be quite different. In this case, assume the budget to diversify the $http$ services cannot exceed \$100K ($q_{http} \leq \$100K$); for ssh, it cannot exceed \$100K ($q_{ftp} \leq \$100$); for

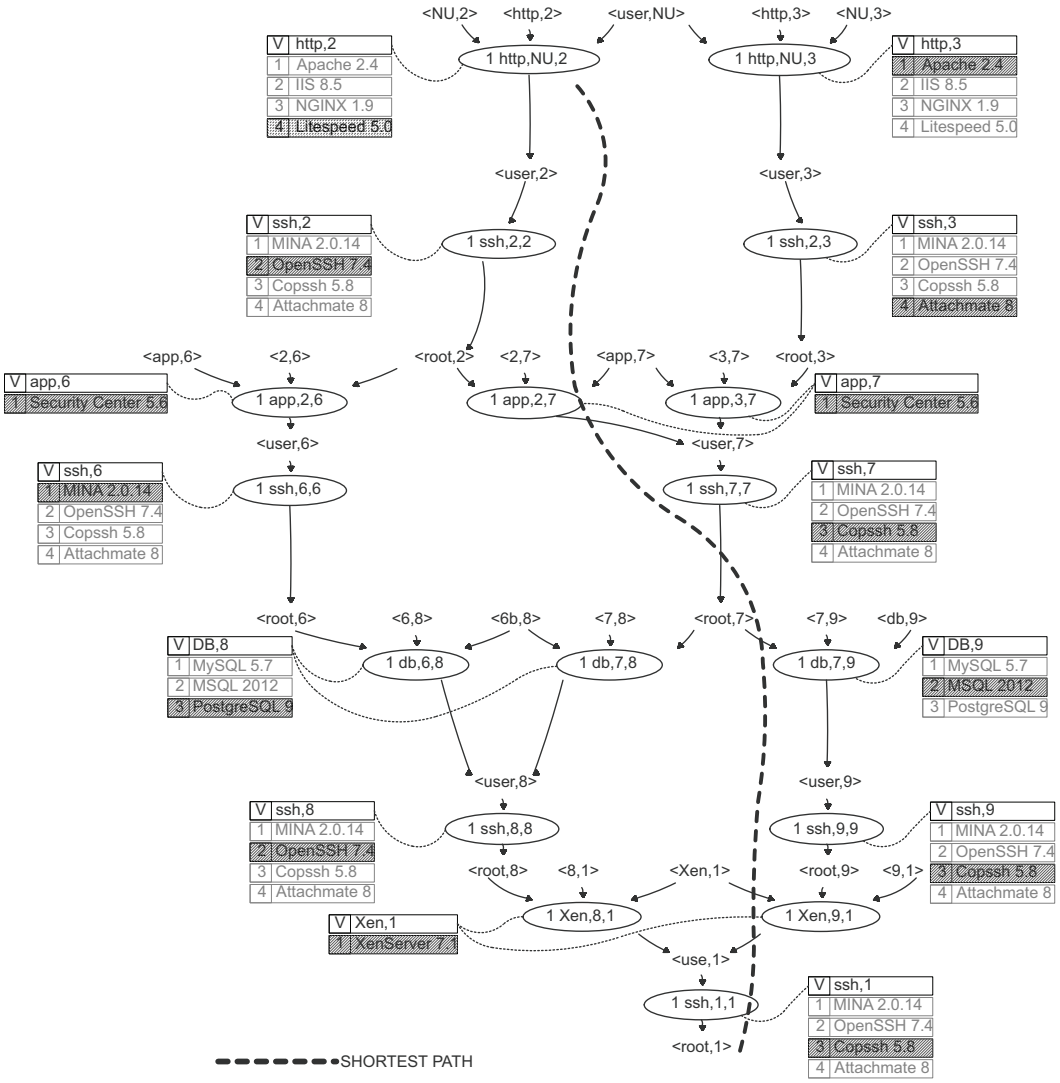


Fig. 7. Use Case A: General and individual budget constraints.

app, it cannot exceed \$25K ($q_{ssh} \leq \$25K$); finally, for db, it cannot exceed \$25K ($q_{smtp} \leq \$25K$). The solution provided by the GA for each one of the diversity metrics are $d_1 = 0.436101$, $d_2 = 3.266839$, and $d3 = 3.01 \times 10^{-8}$. We can see that by enforcing individual budget constraints the network's resilience against zero-day attacks can still be optimized but the additional budget constraints might not allow to achieve the best diversity metric possible.

Use Case B: $Q_d \leq$ \$124K while $q_{http} + q_{ssh} \leq$ \$100K. While use case A shows how individual cost constraints can affect the optimization of the diversity metrics, in practice not all services may be of concern and some may have negligible cost. This use case models such a scenario by assigning a combined budget restriction for only the $http$ and ssh services, i.e., the cost incurred by diversifying these two services should not exceed \$100K.

The solution provided by the GA for each one of the diversity metrics is $d_1 =$

0.436101, $d_2 = 3.266839$, and $d3 = 2.74 \times 10^{-8}$. Here the combined $http/ssh$ budget constraint of \$100K is also satisfied.

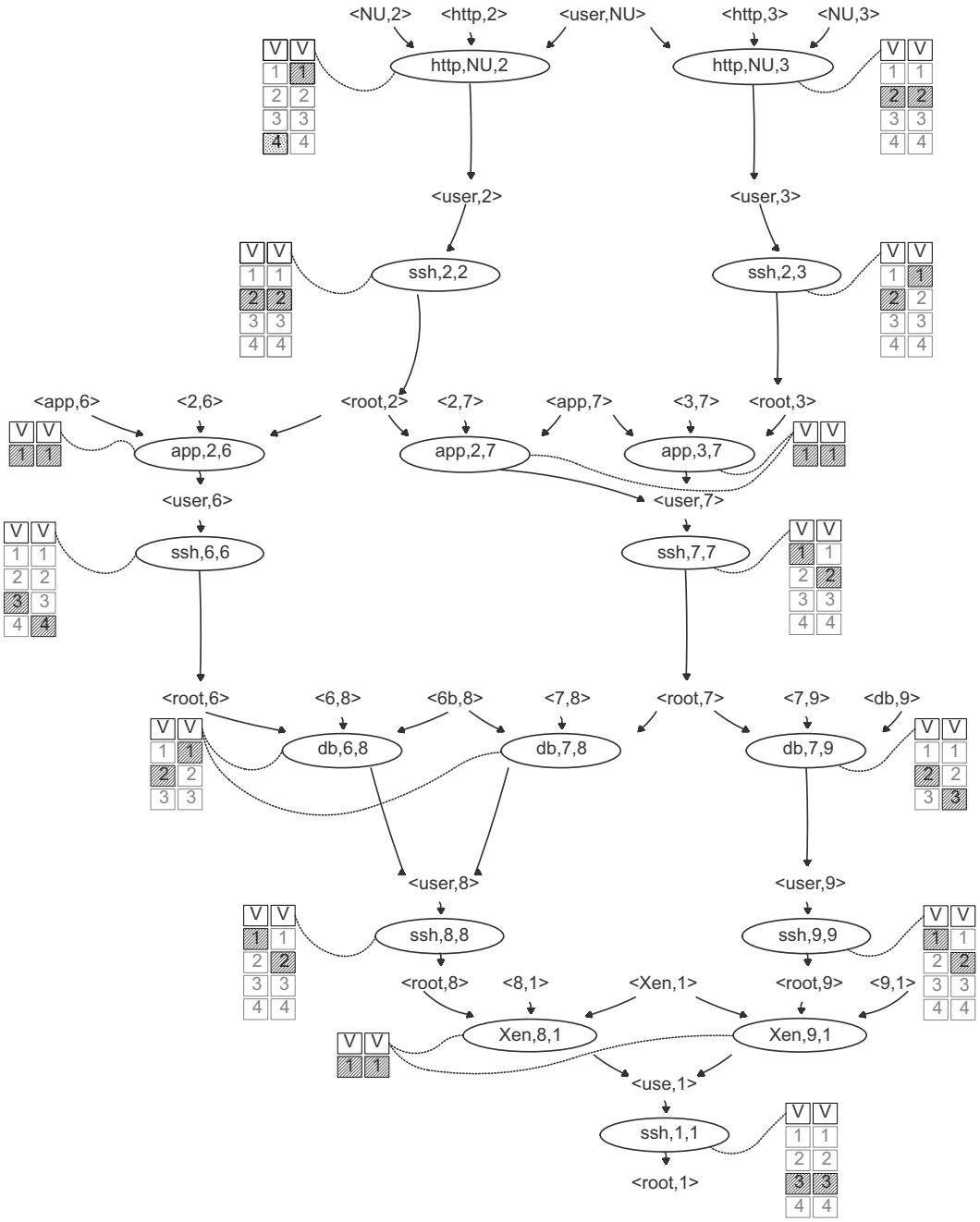


Fig. 8. Use Case B and Use Case C.

Use Case C: $Q_d \leq$ \$300K while $q_{ssh} \leq 0.8 \cdot q_{http}$. This final case deals with scenarios where some services might have a higher priority over others. The constraint in this use case is that the total incurred cost while diversifying the ssh service should not exceed 80% of what is incurred by diversifying the $http$ service.

The solution provided by the GA for each one of the diversity metrics is $d_1 =$

0.527833, $d_2 = 2.828427$, and $d3 = 1.84 \times 10^{-6}$. While the value of the d_1 metric increased to 0.527833, the d_2 and d_3 metrics both decreased. In this case the budget conditions are also satisfied

d-diversity metric value based use case scenarios			
Use Case Scenario	d_1 value	d_2 value	d_3 value
Use Case A	0.555	458	3.464102 3.12×10^{-7}
General + Individual Budgets	0.436	101	3.266839 3.01×10^{-8}
Use Case B	0.436	101	3.266839 2.74×10^{-8}
Use Case C	0.527	833	2.828427 1.84×10^{-6}

Table 2
d-diversity metric values for different use case scenarios.

Based on the results shown in Table 2, it is interesting to note how the different d-diversity metrics behave in response to budget constraints. When certain budget constraints are applied, it is possible that while one d-diversity metric decreases in value (e.g. d_2's value from use case B to use case C), the others could increase (e.g. d_1's value from use case B to C) and vice-versa. This is to be expected as the different d-diversity metrics measure different properties of the network so those metrics might need to be taken into consideration at the same time. Also, as seen from the above use cases, our model and problem formulation makes it relatively straightforward to apply any standard optimization techniques, not limited to GA, in order to optimize the diversity while dealing with different budget constraints.

4.3 Heuristic Algorithm

For all the test cases described above, the calculation of the d_2 diversity metric relies 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 [2,3], it is usually computationally infeasible to enumerate all the available attack paths in a resource graph for large networks. Therefore, we design a heuristic algorithm to reduce the search complexity when calculating d_2, by only storing the m-shortest paths at each step [20], as depicted in Figure 9 and detailed below.

Procedure *Heuristic_m-shortest*
Input: Extended resource graph $\langle G, v\rangle$, critical asset c_g, number of paths m, diversified diversity control vector, D
Output: $\sigma(c_g)$
Method:
1. **Let** $vlist$be any topological sort of G
5. **While** all $vlist$ elements are unprocessed
6. **If** $c \in C_I$ and c is unprocessed
7. **Let** $\sigma(c) = c$
8. **Mark** c as processed
9. **Else if** $e \in E$ (e is not processed) and $(\forall c \in C)((c, e) \in R_r \Rightarrow c$ is processed)
10. **Let** $\{c \in C : (c, e) \in R_r\} = \{c_1, c_2, \ldots, c_n\}$
11. **Let** $\alpha(e) = a_1 \cup a_2 \ldots \cup e : a_i \in \sigma(c_i),\ 1 \leq i \leq n$
13. **Let** $\alpha'(ov(e)) = a'_1 \cup a'_2 \ldots \cup e : a'_i \vdash a_i,\ 1 \leq i \leq n$
12. **If** $n > m$
13. **Let** $\sigma(e) = ShortestM(\langle \alpha(e), \mid Unique(\alpha'[ov(e)]) \mid \rangle, m))$
14. **Else**
15. $\sigma(e) = a_1 \cup a_2 \ldots \cup e : a_i \in \sigma(c_i),\ 1 \leq i \leq m$
16. **Mark** e as processed
17. **Else** (c s.t. $(e, c) \in R_i$ and c is unprocessed)
18. **If** $(\forall e' \in E)((e', c) \in R_i \Rightarrow e'$ is processed)
19. **Let** $\alpha(c) = \bigcup_{e'}$ s.t. $_{(e',c)\in R_i}\ \sigma(e')$
20. **Let** $\alpha'(c) = \bigcup_{e'}$ s.t. $_{(e',c)\in R_i}\ \sigma(ov(e'))$
21. **If** $length(\alpha(c)) > m$
22. **Let** $\sigma(c) = ShortestM(\langle \alpha(c), \mid Unique(\alpha'[ov(c)]) \mid \rangle, m))$
23. **Else**
24. **Let** $\sigma(c) = \bigcup_{e'}$ s.t. $_{(e',c)\in R_i}\ \sigma(e')$
25. **Mark** c as processed
26. **Return** $\sigma(c_g)$

Fig. 9. A Heuristic algorithm for calculating m-shortests paths

The algorithm on Figure 9, which has lineal complexity ($O(N)$), is the one used to check for the m-shortest paths for all three d-diversity metrics. This algorithm starts by topologically sorting the graph (line 1) and proceeds to go through each one of the nodes on the resource graph collection of attack paths, as set of exploits $\sigma()$, that reach that particular node. The main loop cycles through each unprocessed node. If a node is an initial condition, the algorithm assumes that the node itself is the only path to it and it marks it as processed (lines 6-8). For each exploit e, all its preconditions are placed in a set (line 10). The collection of attack paths $\alpha(e)$ is constructed from the attack paths of those preconditions (lines 10 and 11). In a similar way, $\sigma'(ov(e))$ is constructed with the function $ov()$ which, aside of using the exploits includes value of element of the diversity control vector that supervises that exploit.

If there are more than m paths to that node, the algorithm will use the function $Unique$ to first look for unique combinations of service 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 number of unique service and service instance combinations given by each one of the respective paths in $\alpha'(ov(e))$. The function $ShortestM()$ selects the top m keys whose values are the smallest and returns the m paths with the minimum number of distinct combination of services and service instances (line 13). If there are less than m paths, it will return all of the paths (line 15). After this, it marks the node as processed (line 16). The process is similar when going through each one of the intermediate conditions (lines 17-24). Finally,

the algorithm returns the collection of m paths that can reach the critical asset c_g. It is worth noting that the algorithm does not make any distinction in whether or not a particular path has a higher priority over another when they share the same number of unique service/service instance combinations.

5 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 2015b's GA toolbox. To generate many resource graphs for simulations, we first construct a small number of seed graphs based on realistic 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). The resource graphs were used as the input for the optimization toolbox where the objective function is to maximize the diversity metrics d_1 and d_2, and to minimize the d_3 metric, subject to budget constraints.

To determine the genetic operators, we used the hill climbing algorithm. Our simulations show that, 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. Additionally, our experiences also show that, because our largest resource graph had a diversity control vector of fewer than 100 variables, we could set the population size equal to 200; nevertheless, we believe that when dealing with a larger number of optimization variables, the population size should be at least twice the number of variables.

The complexity of our proposed solution will depend on the objective function, the population size, and the length of diversity control vector. While the calculation for d_1 metric is straightforward, the calculation for d_2 is NP-hard since the subproblems of finding the shortest paths (over which the d_2 metric is to be evaluated) in resource graphs is already intractable by the well know results in attack graphs [2,3] and the common syntax between resource graphs and attack graphs. We will therefore rely on the heuristic algorithm presented in Section 4.3 for this purpose. Figure 10 shows that the processing time for d_2 metric 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.

For d_3 diversity metric, calculating the marginal probability with the Bayesian network within the objective function is also an NP-hard optimization problem due to the Cook Levin theorem [44]. For the simulations where the d_3 metric is evaluated, we use OpenBayes. However, because of its well known limitations [45], the com-

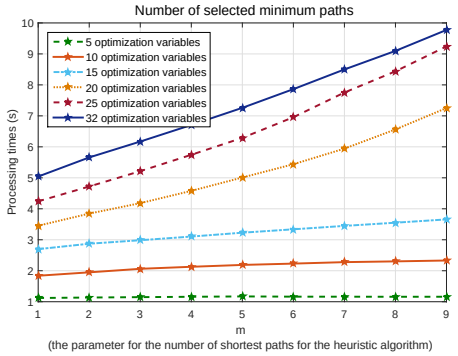


Fig. 10. d_2 Processing time.

putation of d_3 may become prohibitive for large graphs, as shown in Figure 11. To reduce the computational overhead for the optimization of the d_3 metric, we were inspired by [46] to use graphs with cluster structures. The benefit of this approach is that smaller subgraphs (which could represent subnetworks) may be processed separately, which helps to overcome the limitations of OpenBayes to some extent. On the other hand, approximate Bayesian inferences might need to be employed for even larger graphs.

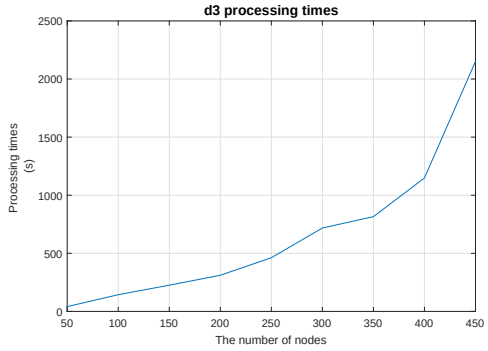


Fig. 11. d_3 Processing times (s).

Because our heuristic algorithm takes only the m-shortest paths to calculate the d_2 diversity metric, the accuracy is an important issue to be considered. Here the accuracy refers to the approximation ratio between the result obtained for the d_2 metric using our heuristic algorithm and that of simply enumerating and searching all the paths while assuming all services and service instances are different ($\frac{d_{Heuristic}}{d_{BruteForce}}$). A ratio close to 1 indicates that our algorithm can provide a solution that is closer to the one provided by enumerating all paths (brute force), and the diversity control vector provided by the GA is used to calculate the accuracy. Figure 12 evaluates the accuracy through simulations. From the results, we can see that when we increase the number of paths considered at each step (m) , the accuracy of the optimized d_2 metric gets closer to the one provided by brute force. We can additionally see that if the m value is greater than or equal to 6, the approximation ratio reaches an acceptable level. For the following simulations, we have settled with an m value of 9.

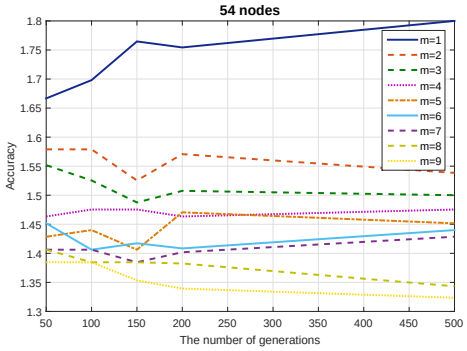


Fig. 12. Accuracy vs m (parameter of the heuristic algorithm).

We also consider the ratio between the difference in the d-diversity metrics before and after optimization, ($\frac{d_{Optimized}-d_{NotOptimized}}{d_{NotOptimized}}$), which will be called the *gain* of the d-diversity metrics (or simply the gain). The gain provides us with an idea on how much room there is to improve the security with respect to given cost constraints using our method. Figure 13 shows the results when the diversity control vector has different numbers of service instances to take from (i.e., different sizes of the service pools). In this simulation, we only consider graphs with a relative high difference in the length of the shortest path before and after all services are diversified using the algorithm.

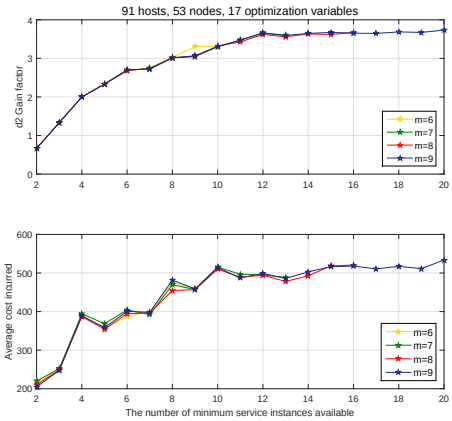


Fig. 13. The effect of available services on a node.

We can see an increasing gain in the d_2 value after optimization, when more service instances are available. However, this trend begins to stall after a certain point. From this observation it can be inferred that the number of available service instances will affect the difference between the maximum d_2 value possible and the minimum d_2, but such an effect also depends on the size of the network (or the extended resource graph), so increasing the number of available service instances does not always help.

Figures 14 and 15 compare the values and the corresponding gain between the d_1 and d_2 metrics. We can see that their average values as well as their average gain

both exhibit a similar trend. Therefore, we will focus more on the d_2 metric from now on.

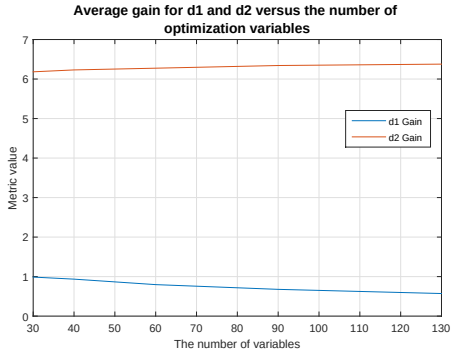


Fig. 14. Average optimized value for d1 and d2 based on number of optimization variables.

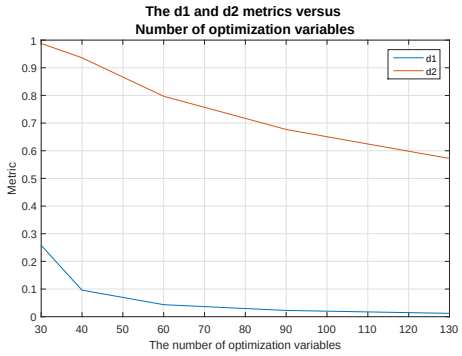


Fig. 15. Average gain of d1 and d2 based on number of optimization variables.

In Figure 16, we analyze the average gain of d_2 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 size of the graphs, the gain will decrease if we keep the same optimization parameters, as well as the same amount of diversifiable services. In Figure 17, we analyze the average gain of the d_3 metric for different sizes of graphs. Like with the d_2 metric, the increase in the number of nodes will reduce the amount of optimization available to the metric.

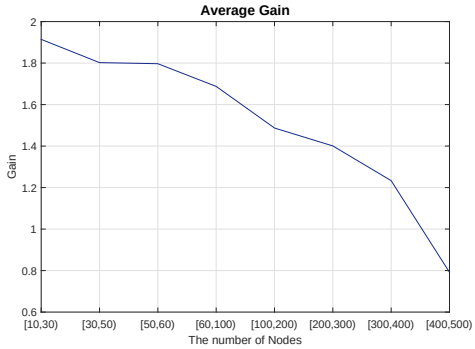


Fig. 16. The average d_2 gain vs the number of nodes.

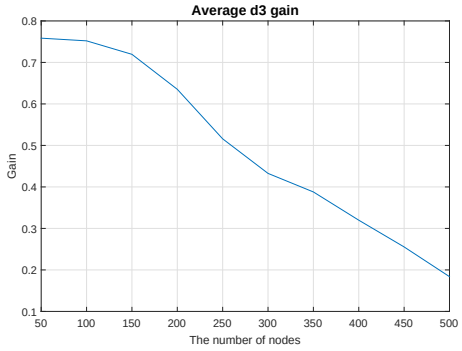


Fig. 17. Average d_3 gain based on the number of nodes.

Figures 18 to 23 show the optimization results for networks with different characteristics (and hence different shapes of resource graphs). First, we examine networks with different levels of protection which is reflected as the relative *depth* of a resource graph. 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 services are diversified. There is a relative linear increase in the gain as we increase the relative distance in the shortest path. While this does not provide an accurate description of the graph's shape, it does provide an idea of how much additional security may be obtained through diversification for different networks, as shown in Figures 18, 19, and 20.

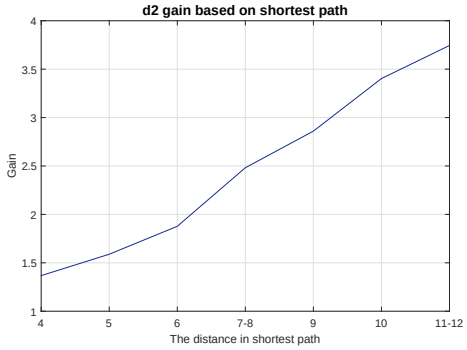


Fig. 18. Average d_2 gain based on relative distance of shortest path.

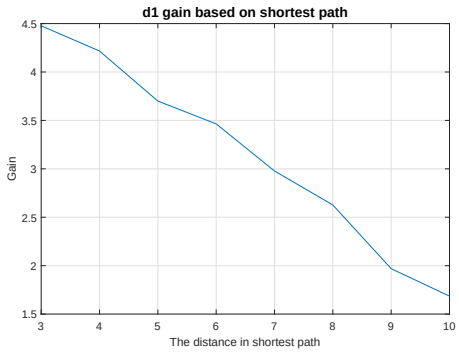


Fig. 19. Average d_1 gain based on relative distance of shortest path.

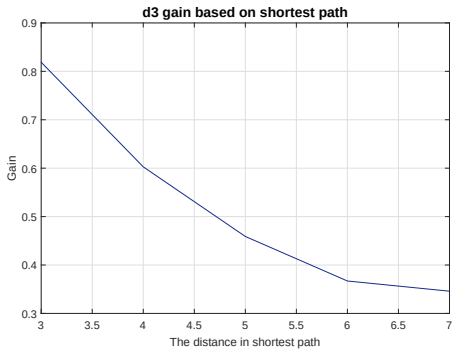


Fig. 20. Average d_3 gain based on relative distance of shortest path.

Finally, in Figures 21, 22, and 23, we can see the effect of the network's *degree of exposure* which is defined as the number of exploits that are initially reachable by the attacker (i.e., the first layer of exploits in resource graphs). As we increase the degree of exposure, the gain in optimization decreases in almost a linear way. That is, there will be less room for diversification if the network is more exposed. Combining those last two sets of results, we can conclude that networks that are already well guarded, in the sense of having a relatively larger depth and a lower degree of exposure, will in fact enjoy more opportunities for further improving the security through service diversification.

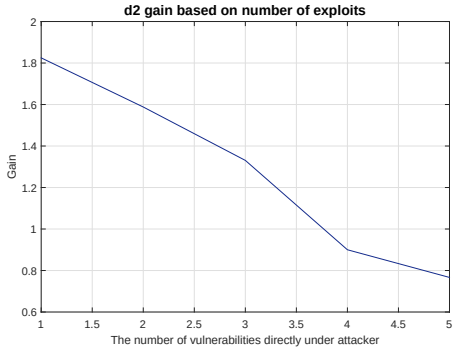


Fig. 21. Average d_2 gain based on directly reachable vulnerabilities.

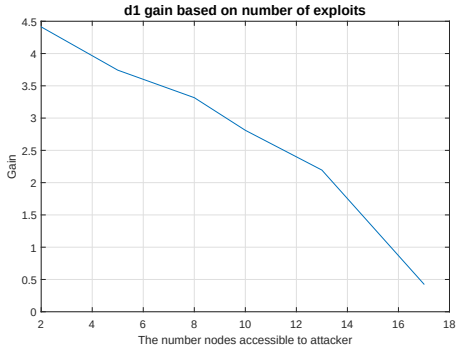


Fig. 22. Average d_1 gain based on directly reachable vulnerabilities.

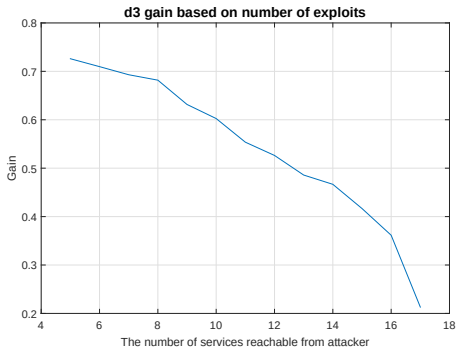


Fig. 23. Average d_3 gain based on directly reachable vulnerabilities.

6 Conclusion

In this paper, we have formulated optimizing the service diversity as an optimization problem using three network diversity metrics, which led to an automated diversity-based network hardening approach against zero-day attacks. This automated approach used a heuristic algorithm that helped to manage the complexity of computing the three diversity metric values as well as limiting the time for optimization to an acceptable level. We have shown some sample cost constraints while our model and formulation would allow for other practical scenarios to be specified and optimized. We have tested the scalability and accuracy of the proposed algorithms through simulation results, and we have also discussed how the gain in the d-diversity metrics value will be affected by the number of available service instances in the service pools and different sizes and shapes of the resource graphs.

We discuss several aspects of the proposed automated optimization technique where additional improvements and evaluations are possible.

- While this paper focuses on diversifying services within a traditional network, a future step is to extend this approach to emerging networks such as SDN-based virtual networks.
- We will consider other optimization algorithms in addition to GA to compare and search for more efficient and effective solutions to our problem.
- Since the three d-diversity metrics measure different network properties, a natural future direction would be to provide multi-objective optimization solutions.

Acknowledgements. Authors with Concordia University are partially supported by the Natural Sciences and Engineering Research Council of Canada under Discovery Grant N01035. Sushil Jajodia was supported in part by the National Institute of Standards and Technology grants 60NANB16D287 and 60NANB18D168, National Science Foundation under grant IIP-1266147, Army Research Office under grant W911NF-13-1-0421, and Office of Naval Research under grant N00014-15-1-2007.

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] A. Avizienis and L. Chen, "On the implementation of n-version programming for software fault tolerance during execution," in *Proc. IEEE COMPSAC*, vol. 77, 1977, pp. 149–155.

[2] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese, "Modeling network diversity for evaluating the robustness of networks against zero-day attacks," in *ESORICS 2014*. Springer, 2014, pp. 494–511.

[3] 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)*, vol. 11, no. 5, pp. 1071–1086, 2016.

[4] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," *Computer Communications*, vol. 29, no. 18, pp. 3812–3824, 2006.

[5] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 204–213.

[6] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Surviving unpatchable vulnerabilities through heterogeneous network hardening options," pp. 1–29, 03 2018.

[7] L. Mieritz and B. Kirwin, "Defining gartner total cost of ownership," *L. Mieritz, B. Kirwin*, 2005.

[8] "Cost of data center outages. data center performance benchmark series," http://www.ponemon.org/blog/2016-cost-of-data-center-outages/, Jan, 2016.

[9] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Diversifying network services under cost constraints for better resilience against unknown attacks," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2016, pp. 295–312.

[10] K. Bakshi, "Cisco cloud computing-data center strategy, architecture, and solutions," *CISCO White Paper. Retrieved October*, vol. 13, p. 2010, 2009.

[11] T. Fifield, D. Fleming, A. Gentle, L. Hochstein, J. Proulx, E. Toews, and J. Topjian, *OpenStack Operations Guide*. " O'Reilly Media, Inc.", 2014.

[12] N. Alhebaishi, L. Wang, S. Jajodia, and A. Singhal, "Threat modeling for cloud data center infrastructures," in *International Symposium on Foundations and Practice of Security*. Springer, 2016, pp. 302–319.

[13] N. Gruschka and M. Jensen, "Attack surfaces: A taxonomy for attacks on cloud services," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 276–279.

[14] J. Barr, "Building three-tier architectures with security groups," https://aws.amazon.com/blogs/aws/building-three-tier-architectures-with-security-groups/, June, 2010.

[15] R. Dewri, I. Ray, N. Poolsappasit, and D. Whitley, "Optimal security hardening on attack tree models of networks: a cost-benefit analysis," *International Journal of Information Security*, vol. 11, no. 3, pp. 167–188, 2012.

[16] I. Ray and N. Poolsapassit, "Using attack trees to identify malicious attacks from authorized insiders," in *ESORICS 2005*. Springer, 2005, pp. 231–246.

[17] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, 2002, pp. 217–224.

[18] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 273–284.

[19] L. Wang, A. Singhal, and S. Jajodia, "Measuring the overall security of network configurations using attack graphs," in *Data and Applications Security XXI*. Springer, 2007, pp. 98–112.

[20] M. Albanese, S. Jajodia, and S. 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*. IEEE, 2012, pp. 1–12.

[21] L. Wang, S. Jajodia, A. Singhal, and S. Noel, "k-zero day safety: Measuring the security risk of networks against unknown attacks," in *ESORICS 2010*. Springer, 2010, pp. 573–587.

[22] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *Dependable and Secure Computing, IEEE Transactions on*, vol. 11, no. 1, pp. 30–44, 2014.

[23] J. McHugh, "Quality of protection: measuring the unmeasurable?" in *Proceedings of the 2nd ACM workshop on Quality of protection*. ACM, 2006, pp. 1–2.

[24] L. Wang, M. Albanese, and S. Jajodia, *Network Hardening: An Automated Approach to Improving Network Security*. Springer Publishing Company, Incorporated, 2014.

[25] M. Gupta, J. Rees, A. Chaturvedi, and J. Chi, "Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach," *Decision Support Systems*, vol. 41, no. 3, pp. 592–603, 2006.

[26] S. Wang, Z. Zhang, and Y. Kadobayashi, "Exploring attack graph for cost-benefit security hardening: A probabilistic approach," *Computers & security*, vol. 32, pp. 158–169, 2013.

[27] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using bayesian attack graphs," *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 1, pp. 61–74, 2012.

[28] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 62:1–62:35, Dec. 2016. [Online]. Available: http://doi.acm.org/10.1145/3005714

[29] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *Security & Privacy, IEEE*, vol. 4, no. 6, pp. 85–89, 2006.

[30] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2008, pp. 283–296.

[31] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic bayesian network," in *Proceedings of the 4th ACM workshop on Quality of protection*. ACM, 2008, pp. 23–30.

[32] "Nist special publication 500-307: Cloud computing service metrics description (2015)," https://www.nist.gov/sites/default/files/documents/itl/cloud/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf, Accessed September, 2017.

[33] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-variant systems: a secretless framework for security through diversity," in *Usenix Security*, vol. 6, 2006, pp. 105–120.

[34] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance measurement using hidden markov models," in *Recent Advances in Intrusion Detection*. Springer, 2006, pp. 19–40.

[35] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Os diversity for intrusion tolerance: Myth or reality?" in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*. IEEE, 2011, pp. 383–394.

[36] A. Gunasekaran, *Organizational Advancements through Enterprise Information Systems: Emerging Applications and Developments: Emerging Applications and Developments*. IGI Global, 2009.

[37] "Amazon.com, inc. revenue and earnings per share (eps)," http://www.nasdaq.com/symbol/amzn/revenue-eps/, June, 2017.

[38] "Number of amazon.com employees from 2007 to 2016," https://www.statista.com/statistics/234488/number-of-amazon-employees/, June, 2017.

[39] S. Jajodia, S. Noel, and B. O'Berry, "Topological analysis of network attack vulnerability," in *Managing Cyber Threats: Issues, Approaches and Challenges*, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. Kluwer Academic Publisher, 2003.

[40] P. Festa, "A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems," in *Transparent Optical Networks (ICTON), 2014 16th International Conference on*. IEEE, 2014, pp. 1–20.

[41] H. M. Azamathulla, F.-C. Wu, A. Ab Ghani, S. M. Narulkar, N. A. Zakaria, and C. K. Chang, "Comparison between genetic algorithm and linear programming approach for real time operation," *Journal of Hydro-environment Research*, vol. 2, no. 3, pp. 172–181, 2008.

[42] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, 1989.

[43] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[44] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*. ACM, 1971, pp. 151–158.

[45] M. Arias and F. J. Dıez, "Carmen: An open source project for probabilistic graphical models," in *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM08)*, 2008, pp. 25–32.

[46] L. Muñoz-González, D. Sgandurra, A. Paudice, and E. C. Lupu, "Efficient attack graph analysis through approximate inference," *arXiv preprint arXiv:1606.07025*, 2016.