Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks

Mengyuan Zhang, Lingyu Wang, *Member, IEEE*, Sushil Jajodia, *Fellow, IEEE*, Anoop Singhal, *Senior Member, IEEE*, and Massimiliano Albanese, *Member, IEEE*

***Abstract*— Diversity has long been regarded as a security mechanism for improving the resilience of software and networks against various attacks. More recently, diversity has found new applications in cloud computing security, moving target defense, and improving the robustness of network routing. However, most existing efforts rely on intuitive and imprecise notions of diversity, and the few existing models of diversity are mostly designed for a single system running diverse software replicas or variants. At a higher abstraction level, as a global property of the entire network, diversity and its effect on security have received limited attention. In this paper, we take the first step toward formally modeling network diversity as a security metric by designing and evaluating a series of diversity metrics. In particular, we first devise a biodiversity-inspired metric based on the effective number of distinct resources. We then propose two complementary diversity metrics, based on the least and the average attacking efforts, respectively. We provide guidelines for instantiating the proposed metrics and present a case study on estimating software diversity. Finally, we evaluate the proposed metrics through simulation.**

***Index Terms*— Biodiversity, computer security, firewalls, information security, intrusion detection.**

I. Introduction

PROTECTING mission critical computer networks, such as those used in critical infrastructures and military organizations, demands more than just patching known vulnerabilities and deploying firewalls or IDSs. Improving the resilience of such networks against potential zero day attacks exploiting unknown vulnerabilities is equally important, which is evidenced by the fact that modern malware may exploit multiple unknown vulnerabilities at the same time [1]. However, dealing with unknown vulnerabilities is clearly a challenging task. Although we already have many effective solutions (e.g., IDS/IPS, firewalls, antivirus, software upgrading and patching) for tackling known attacks, zero day attacks are known to be difficult to mitigate due to the lack of information.

To this end, diversity has long been regarded as a valuable solution because it may improve the resilience of a software system against both known and unknown vulnerabilities [2] (a more detailed review of related work will be given in Section VIII). Security attacks exploiting unknown vulnerabilities may be detected and tolerated as Byzantine faults by comparing either the outputs [3] or behaviors [4] of multiple software replicas or variants [5]. Although the earlier diversity-by-design approaches usually suffer from prohibitive development and deployment cost, recent works show more promising results on employing either opportunistic diversity [6] or automatically generated diversity [7]–[9]. More recently, diversity has found new applications in cloud computing security [10], Moving Target Defense (MTD) [11], resisting sensor worms [12], and network routing [13]. Most of those existing efforts rely on either intuitive notions of diversity or models mostly designed for a single system running diverse software replicas or variants.

However, at a higher abstraction level, as a global property of an entire network, the concept of *network diversity* and its effect on security has received limited attention. In this paper, we take the first step towards formally modeling network diversity as a security metric for the purpose of evaluating the resilience of networks with respect to zero day attacks. More specifically,

- First, we propose a network diversity metric by adapting well known mathematical models of biodiversity in ecology. The metric basically counts the number of distinct resources inside a network, while considering the uneven distribution of resources and varying degree of similarity between resources. This first metric is suitable for cases where all the resources are regarded as equally important, and it can also serve as a building block of other metrics. The main limitation is that it ignores potential causal relationships between resources in a network.

Manuscript received June 22, 2015; revised October 1, 2015 and December 17, 2015; accepted December 20, 2015. Date of publication January 12, 2016; date of current version February 24, 2016. This work was supported in part by the National Science Foundation under Grant IIP-1266147, in part by the Natural Sciences and Engineering Research Council of Canada Discovery under Grant N01035, in part by the National Institute of Standards and Technology under Grant 60NANB14D060 and Grant 60NANB15D091, in part by the Office of Naval Research under Grant N00014-15-1-2007, and in part by the Army Research Office under Grant W911NF-13-1-0421. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jianying Zhou.

M. Zhang and L. Wang are with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: mengy_zh@ciise.concordia.ca; wang@ciise.concordia.ca).

S. Jajodia and M. Albanese are with the Center for Secure Information Systems, George Mason University, Fairfax, VA 22030 USA (e-mail: jajodia@gmu.edu; malbanes@gmu.edu).

A. Singhal is with the Computer Security Division, National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (e-mail: anoop.singhal@nist.gov).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2516916

- Second, we design a network diversity metric based on the least attacking effort required for compromising certain important resources, while taking into account the causal relationships between resources. This metric is suitable for cases where administrators are mostly concerned about some critical assets (e.g., storage or database servers). However, by focusing on the least attacking effort, the metric only provides a partial picture about how diversity may affect security.
- Third, we devise a probabilistic network diversity metric to reflect the average attacking effort required for compromising critical assets. This metric serves as a complementary measure to the above second metric in depicting the effect of diversity on security.
- Fourth, we provide guidelines for instantiating the proposed metric models for given networks. In particular, we demonstrate how software similarity may be estimated through a case study on different versions of Chrome.
- Finally, we evaluate and compare the three metrics through simulation results under different use cases.

The main contribution of this paper is as follows. To the best of our knowledge, this is the first effort on systematically modeling network diversity as a security metric. As we will demonstrate shortly, an intuitive notion of diversity can usually cause misleading results, whereas our formal model of network diversity will enable a better understanding of the effect of diversity on security. Our work is also among the first efforts on borrowing the biodiversity concepts from ecology and applying it to network security, and we believe this initial effort may generate further interest in this direction. Finally, the guidelines for instantiating the models and related case studies and simulation results will ease the transition from theoretical results to practical solutions based on the proposed metrics.

The preliminary version of this paper has previously appeared in [14]. This paper has substantially improved and extended the previous version. The most significant extensions include a new probabilistic model for addressing various limitations of the previous model appearing in [14] (Section IV), guidelines for instantiating the metrics and a case study (Section VII), discussions on software similarity (Section VII), and finally a series of simulations for characterizing the proposed metrics (Section VI).

The rest of this paper is organized as follows. Section II presents use cases and the biodiversity-inspired metric. Section III and Section IV propose the least and average attacking effort-based metrics, respectively. Section V discusses how to instantiate the metric models and presents a case study. Section VI gives simulation results and discussion about the software similarity. Section VII discusses software similarity. Section VIII reviews related work, and finally Section IX discusses limitations and gives some concluding remarks.

II. Preliminaries

This section presents several use cases and defines a biodiversity-inspired network diversity metric.

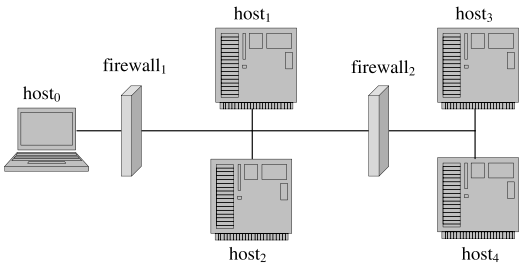


Fig. 1. The running example.

A. Use Cases

We describe several use cases in order to motivate our study and illustrate various requirements and challenges in modeling network diversity. Some of those use cases will also be revisited in later sections.

1) Use Case 1 (Stuxnet and SCADA Security): Stuxnet is one of the first malware that employ multiple (four) different zero day attacks [1]. This clearly indicates that, in a mission critical system, such as supervisory control and data acquisition (SCADA) in this case, the risk of zero day attacks and multiple unknown vulnerabilities is very real, and consequently network administrators will need a systematic way for evaluating such a risk. However, this is clearly a challenging task due to the lack of prior knowledge about vulnerabilities or attacking methods.

A closer look at Stuxnet's attack strategies will reveal how network diversity may help here. Stuxnet targets the programmable logic controllers (PLCs) on control systems of gas pipelines or power plants [1], which are mostly programmed using Windows machines not connected to the network. Therefore, Stuxnet adopts a multi-stage approach, by first infecting Windows machines owned by third parties (e.g., contractors), next spreading to internal Windows machines through the LAN, and finally covering the last hop through removable flash drives [1]. Clearly, the degree of software diversity along potential attack paths leading from the network perimeter to the PLCs can be regarded as a critical metric of the network's resilience against a threat like Stuxnet. Our objective in this paper is to provide a rigorous study of such network diversity metrics.

2) Use Case 2 (Worm Propagation): To make our discussion more concrete, we will refer to the running example shown in Figure 1 from now on. In this use case, our main concern is the potential propagation of worms or bots inside the network. A common belief here is that we can simply count the number (percentage) of distinct resources in the network as diversity. Although such a definition is natural and intuitive, it clearly has limitations.

For example, suppose host 1, 2, and 3 are Web servers running IIS, all of which access files stored on host 4. Clearly, the above count-based metric will indicate a lack of diversity and suggest replacing IIS with other software to prevent a worm from infecting all three at once. However, it is easy to see that, even if a worm can only infect one Web server after such a diversification effort (e.g,. it can infect IIS but not Apache), it can still propagate to all four hosts through the

network share on host 4 (e.g., it may infect certain executable files stored on host 4 which are subsequently accessed by all Web servers). The reason that this naive approach fails in this case is that it ignores the existence of causal relationships between resources (due to the network share). Therefore, after we discuss the count-based metric in Section II-B, we will address this limitation with a *goal oriented* approach in Section III.

3) Use Case 3 (Targeted Attack): Suppose now we are more concerned with a targeted attack on the storage server, host 4. Following above discussions, an intuitive solution is to diversify resources along any *path* leading to the critical asset (host 4), e.g., between hosts 1 (or 2, 3) and host 4. Although this is a valid observation, realizing it requires a rigorous study of the causal relationships between different resources, because host 4 is only as secure as the weakest path (representing the least attacking effort) leading to it. We will propose a formal metric based on such an intuition in Section III.

On the other hand, the least attacking effort by itself only provides a partial picture. Suppose now host 1 and 2 are diversified to run IIS and Apache, respectively, and firewall 2 will only allow host 1 and 2 to reach host 4. Although the least attacking effort has not changed, this diversification effort has actually provided attackers more opportunities to reach host 4 (by exploiting either IIS or Apache). That is, misplaced diversity may in fact hurt security. This will be captured by a probabilistic metric in Section IV.

4) Use Case 4 (MTD): Moving Target Defense (MTD) can be considered as a different approach to applying diversity to security, since it diversifies resources along the time dimension [11]. However, most existing work on MTD relies on intuitive notion of diversity which may lead to misleading results. This next case demonstrates the usefulness of our proposed metrics particularly for MTD. In this case, suppose host 1 and 2 are Web servers, host 3 an application server, and host 4 a database server. A MTD will attempt to achieve better security by varying in time the software components at different tiers. A common misconception here is that the combination of different components at different tiers will increase diversity, and the degree of diversity is equal to the product of diversity at those tiers. However, this is usually not the case. For example, a single flaw in the application server (host 3) may result in a SQL injection that compromises the database server (host 4) and consequently leaks the root user's password. Also, similar to the previous case, more diversity over time may actually provide attackers more opportunities to find flaws. The lesson here is again that, an intuitive observation may be misleading, and formally modeling network diversity is necessary.

B. Biodiversity-Inspired Network Diversity Metric

Although the notion of network diversity has attracted limited attention, its counterpart in ecology, *biodiversity*, and its positive impact on the ecosystem's stability has been investigated for many decades [15]. While many lessons may potentially be borrowed from the rich literature of biodiversity, in this paper we will focus on adapting existing mathematical models of biodiversity for modeling network diversity.

Specifically, the number of different species in an ecosystem is known as *species richness* [16]. Similarly, given a set of distinct resource types[1] R in a network, we call the cardinality $\mid R \mid$ the *richness* of resources in the network. An obvious limitation of this richness metric is that it ignores the relative abundance of each resource type. For example, the two sets $\{r_1, r_1, r_2, r_2\}$ and $\{r_1, r_2, r_2, r_2\}$ have the same richness of 2 but clearly different levels of diversity.

To address this limitation, the Shannon-Wiener index, which is essentially the Shannon entropy using natural logarithm, is used as a *diversity index* to group all systems with the same level of diversity, and the exponential of the diversity index is regarded as the *effective number* metric [17]. The effective number basically allows us to always measure diversity in terms of the number of equally-common species, even if in reality those species may not be equally common. In the following, we borrow this concept to define the effective resource richness and our first diversity metric.

Definition 1 (Effective Richness and d_1-Diversity): In a network G with the set of hosts $H = \{h_1, h_2, \ldots, h_n\}$, set of resource types $R = \{r_1, r_2, \ldots, r_m\}$, and the resource mapping $res(.) : H \rightarrow 2^R$ (here 2^R denotes the power set of R), let $t = \sum_{i=1}^{n} \mid res(h_i) \mid$ (the total number of resource instances), and finally let $p_j = \frac{|\{h_i : r_j \in res(h_i)\}|}{t} (1 \leq i \leq n, 1 \leq j \leq m)$ (the relative frequency of each resource). We define the network's diversity as $d_1 = \frac{r(G)}{t}$, where $r(G)$ is the network's effective richness of resources, defined as

$$r(G) = \frac{1}{\prod_1^n p_i^{p_i}}$$

One limitation of the effective number-based metric is that similarity between different resource types is not taken into account and all resource types are assumed to be entirely different, which is not realistic (e.g., the same application can be configured to fulfill totally different roles, such as NGinx as a reverse proxy or a web server, respectively, in which case these should be regarded as different resources with high similarity). Therefore, we borrow the similarity-sensitive biodiversity metric recently introduced in [18] to redefine resource richness. With this new definition, the above diversity metric d_1 can now handle similarity between resources.

Definition 2 (Similarity-Sensitive Richness): With respect to Definition 1, suppose a similarity function is given as $z(.) : [1, m] \times [1, m] \rightarrow [0, 1]$ (a larger value denoting higher similarity and $z(i, i) = 1$ for all $1 \leq i \leq m$), and let $zp_i = \sum_{j=1}^{m} z(i, j) p_j$. We define the network's effective richness of resources, considering the similarity function, as

$$r(G) = \frac{1}{\prod_1^n zp_i^{p_i}}$$

The effective richness-based network diversity metric d_1 is only suitable for cases where all resources may be treated equally, and causal relationships between resources either do

[1]We will consider similarity between resources later.

TABLE I
ATTACK PATHS

Attack Path	# of Steps	# of Resources
1. $\langle http, 0, 1\rangle \rightarrow \langle ssh, 1, 4\rangle \rightarrow \langle rsh, 4, 5\rangle$	3	3
2. $\langle http, 0, 1\rangle \rightarrow \langle ssh, 1, 4\rangle \rightarrow \langle http, 4, 5\rangle$	3	2
3. $\langle http, 0, 1\rangle \rightarrow \langle http, 1, 2\rangle \rightarrow \langle ssh, 2, 4\rangle \rightarrow \langle rsh, 4, 5\rangle$	4	3
4. $\langle http, 0, 1\rangle \rightarrow \langle http, 1, 2\rangle \rightarrow \langle ssh, 2, 4\rangle \rightarrow \langle http, 4, 5\rangle$	4	2
5. $\langle firewall, 0, F\rangle \rightarrow \langle http, 0, 2\rangle \rightarrow \langle ssh, 2, 4\rangle \rightarrow \langle rsh, 4, 5\rangle$	4	4
6. $\langle firewall, 0, F\rangle \rightarrow \langle http, 0, 2\rangle \rightarrow \langle ssh, 2, 4\rangle \rightarrow \langle http, 4, 5\rangle$	4	3

not exist or may be safely ignored. On the other hand, this metric may also be used as a building block inside other network diversity metrics, in the sense that we may simply say "the number of distinct resources" without worrying about uneven distribution of resource types or similarity between resources, thanks to the effective richness concepts given in Definition 1 and 2.

The effect of biodiversity on the stability of an ecosystem has been shown to critically depend on the interaction of different species within a food Web [19]. Although such interaction typically takes the form of a "feed-on" relationship between different species, which does not directly apply to computer networks, this observation has inspired us to model diversity based on the structural relationship between resources, which will be detailed in the coming sections.

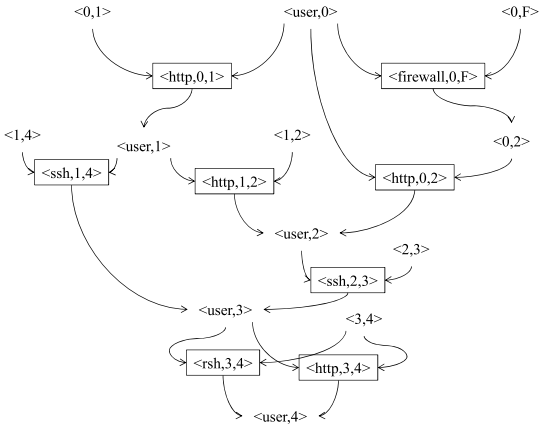


Fig. 2. An example resource graph.

III. LEAST ATTACKING EFFORT-BASED NETWORK DIVERSITY METRIC

This section models network diversity based on the least attacking effort. Section III-A defines the metric, and Section III-B discusses the complexity and algorithm.

A. The Model

In order to model diversity based on the least attacking effort while considering causal relationships between different resources, we first need a model of such relationships and possible zero day attacks. Our model is similar to the *attack graph* model [20], [21], although our model focuses on remotely accessible resources (e.g., services or applications that are reachable from other hosts in the network), which will be regarded as placeholders for potential zero day vulnerabilities instead of known vulnerabilities as in attack graphs.

To build intuitions, we revisit Figure 1 by making the following assumptions. Access from outside firewall 1 is allowed to host 1 but blocked to host 2; accesses from host 1 or 2 are allowed to host 3 but blocked to host 4 by firewall 2; hosts 1 and 2 provide $http$ service; host 3 provides ssh service; host 4 provides both $http$ and rsh services.

Figure 2 depicts a corresponding *resource graph*, which is syntactically equivalent to an attack graph, but models zero day attacks rather than known vulnerabilities. Each pair in plaintext is a self-explanatory security-related condition (e.g., connectivity $\langle source, destination\rangle$ or privilege $\langle privilege, host\rangle$), and each triple inside a box is a potential exploit of resource $\langle resource,\ source\,host, destination\,host\rangle$; the edges point from the pre-conditions to a zero day exploit (e.g., from $\langle 0, 1\rangle$ and $\langle user, 0\rangle$ to $\langle http, 0, 1\rangle$), and from that exploit to its post-conditions (e.g., from $\langle http, 0, 1\rangle$ to $\langle user, 1\rangle$). Exploits or conditions involving firewall 2 are omitted for simplicity.

We simply regard resources of different types as entirely different (their similarity can be handled using the effective resource richness given in Definition 2). Also, we take the conservative approach of considering all resources (services and firewalls) to be potentially vulnerable to zero day attacks. Definition 3 formally introduces the concept of resource graph.

Definition 3 (Resource Graph): Given a network with the set of hosts H, set of resources R with the resource mapping $res(.) : H \rightarrow 2^R$, set of zero day exploits $E = \{\langle r, h_s, h_d\rangle \mid h_s \in H, h_d \in H, r \in res(h_d)\}$ and their pre- and post-conditions C, a resource graph is a directed graph $G(E \cup C, R_r \cup R_i)$ where $R_r \subseteq C \times E$ and $R_i \subseteq E \times C$ are the pre- and post-condition relations, respectively.

Next consider how attackers may potentially attack a critical network asset, modeled as a goal condition, with the least effort. In Figure 2, by following the simple rule that an exploit may be executed if all the pre-conditions are satisfied, and executing that exploit will cause all the post-conditions to be satisfied, we may observe six *attack paths*, as shown in Table I (the second and third columns can be ignored for now and will be explained shortly). Definition 4 formally introduces the concept of attack path.

Definition 4 (Attack Path): Given a resource graph $G(E \cup C, R_r \cup R_i)$, we call $C_I = \{c : c \in C, (\nexists e \in E)(\langle e, c\rangle \in R_i)\}$ the set of initial conditions. Any sequence of zero day exploits $e_1, e_2, \ldots, e_n$ is called an attack path in G, if $(\forall i \in [1, n])(\langle c, e_i\rangle \in R_r \rightarrow (c \in C_i \vee (\exists j \in [1, i-1])(\langle e_j, c\rangle \in R_i)))$, and for any $c \in C$, we use $seq(c)$ for the set of attack paths $\{e_1, e_2, \ldots, e_n : \langle e_n, c\rangle \in R_i\}$.

We are now ready to consider how diversity could be defined based on the least attacking effort (the shortest path). There are actually several possible ways for choosing such shortest paths and for defining the metric, as we will illustrate through our running example in the following.

- First, as shown in the second column of Table I, path 1 and 2 are the shortest in terms of the *steps* (i.e., the number of zero day exploits). Clearly, those do not reflect the least attacking effort, since path 4 may actually take less effort than path 1, as attackers may reuse their exploit code, tools, and skills while exploiting the same $http$ service on three different hosts.
- Next, as shown in the third column, path 2 and 4 are the shortest in terms of the number of distinct resources (or effective richness). This seems more reasonable since it captures the saved effort in reusing exploits. However, although path 2 and 4 have the same number of distinct resources (2), they clearly reflect different diversity.
- Another seemingly valid solution is to base on the minimum ratio $\frac{\#\,of\ resources}{\#\,of\ steps}$ (which is given by path 4 in this example), since such a ratio reflects the potential improvements in terms of diversity (e.g., the ratio $\frac{2}{4}$ of path 4 indicates 50% potential improvement in diversity). However, we can easily imagine a very long attack path minimizing such a ratio but not reflecting the least attacking effort (e.g., an attack path with 9 steps and 3 distinct resources will yield a ratio of $\frac{1}{3}$, less than $\frac{2}{4}$, but clearly requires more effort than path 4).
- Finally, yet another option is to choose the shortest path that minimizes both the number of distinct resources (path 2 and 4) and the above ratio $\frac{\#\,of\ resources}{\#\,of\ steps}$ (path 4). However, a closer look will reveal that, although path 4 does represent the least attacking effort, it does not represent the maximum amount of potential improvement in diversity, because once we start to diversify path 4, the shortest path may change to be path 1 or 2.

Based on these observations, we define network diversity by combining the first two options above. Specifically, the network diversity is defined as the ratio between the minimum number of distinct resources on a path and the minimum number of steps on a path (note these can be different paths). Going back to our running example above, we find path 2 and 4 to have the minimum number of distinct resources (two), and also path 1 and 2 to have the minimum number of steps (three), so the network diversity in this example is equal to $\frac{2}{3}$ (note that it is a simple fact that this ratio will never exceed 1). Intuitively, the numerator 2 denotes the network's current level of robustness against zero day exploits (no more than 2 different attacks), whereas the denominator 3 denotes the network's maximum potential of robustness (tolerating no more than 3 different attacks) by increasing the amount of diversity (from $\frac{2}{3}$ to 1). More formally, we introduce our second network diversity metric in Definition 5 (note that, for simplicity, we only consider a single goal condition for representing the given critical asset, which is not a limitation since multiple goal conditions can be easily handled through adding a few dummy conditions [22]).

Definition 5 (d_2-Diversity): Given a resource graph $G(E \cup C, R_r \cup R_i)$ and a goal condition $c_g \in C$, for each $c \in C$ and $q \in seq(c)$, denote $R(q)$ for $\{r : r \in R, r \text{ appears in } q\}$, the network diversity is defined as (where $min(.)$ returns the minimum value in a set)

$$d_2 = \frac{min_{q \in seq(c_g)} \mid R(q) \mid}{min_{q' \in seq(c_g)} \mid q' \mid}.$$

B. The Complexity and Algorithm

Since the problem of finding the shortest paths (in terms of the number of exploits) in an attack graph (which is syntactically equivalent to a resource graph) is known to be intractable [20], not surprisingly, the problem of determining the network diversity d_2 is also intractable, as stated in Theorem 1 (the proof is omitted and can be found in [14]).

Theorem 1: Given a resource graph $G(E \cup C, R_r \cup R_i)$, determining the network diversity d_2 is NP-hard.

Although determining d_2 in general is computationally infeasible, it may be estimated within a reasonable time using heuristics. In particular, we have proposed an algorithm that employs the heuristic of only maintaining a limited number of local optima at each step in order to keep the complexity manageable [14]. The algorithm is shown to run in polynomial time under the assumption that each exploit has a constant number of pre- and post-conditions. The simulation results also confirm that the algorithm provides accurate enough estimation of d_2 in an acceptable amount of time.

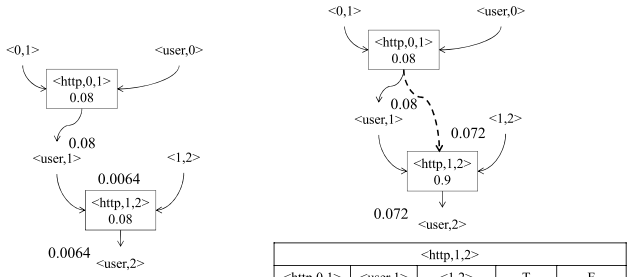
IV. Probabilistic Network Diversity

In this section, we develop a probabilistic metric to capture the effect of diversity based on average attacking effort by combining all attack paths. The preliminary version of this paper [14] has proposed a probabilistic metric model for this purpose. We will first identify important limitations in this model, and then provide a redesigned model to address them.

A. Overview

The model introduced in [14] defines network diversity as the ratio between two probabilities, namely, the probability that given critical assets may be compromised, and the same probability but with an additional assumption that all resource instances are distinct (which means attackers cannot reuse any exploit). Both probabilities represent the *attack likelihood* with respect to goal conditions, which can be modeled using a Bayesian network constructed based on the resource graph [23].

For example, Figure 3 demonstrates this model based on our running example (only part of the example is shown for simplicity). The left-hand side represents the case in which the effect of reusing an exploit is not considered, that is, the two $http$ service instances are assumed to be distinct. The right-hand side considers that effect and models it as the conditional probability that the lower $http$ service may be exploited given that the upper one is already exploited (represented using a dotted line). The two conditional probability tables (CPTs) illustrate the effect of reusing the $http$ exploit



<http,1,2>			
<user,1>	<1,2>	T	F
T	T	0.08	0.92
T	F	0	1
F	T	0	1
F	F	0	1

<http,1,2>				
<http,0,1>	<user,1>	<1,2>	T	F
T	T	T	0.9	0.1
T	T	F	0	1
T	F	T	0	1
F	T	T	0	1
F	T	F	0	1
F	F	F	0	1

Fig. 3. Modeling network diversity using Bayesian networks.

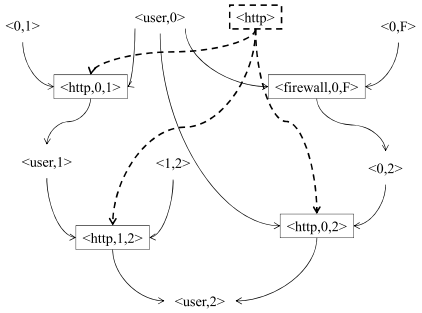


Fig. 4. The redesigned model.

(e.g., probability 0.9 in the right CPT), and not reusing it (e.g., probability 0.08 in the left CPT), respectively. The network diversity in this case will be calculated as the ratio $d_3 = \frac{0.0064}{0.072}$.

We realized that the above model has certain limitations when a few invalid results (larger than 1) were returned during our simulations. More specifically, in the above model, modeling the effect of reusing exploits as a conditional probability (that a resource may be exploited given that some other instances of the same type are already exploited) essentially assumes a total order over different instances of the same resource type in any resource graph, which comprises a major limitation. For example, in Figure 4,[2] although the reused *http* exploit $\langle http, 1, 2\rangle$ (after exploiting $\langle http, 0, 1\rangle$) may be handled using the above model by adding a dotted line pointing to it from its ancestor $\langle http, 0, 1\rangle$, the same method will not work for the other potentially reused *http* exploit $\langle http, 0, 2\rangle$, since there does not exist a definite order between $\langle http, 0, 1\rangle$ and $\langle http, 0, 2\rangle$, which means an attacker may reach $\langle http, 0, 2\rangle$ before, or after, reaching $\langle http, 0, 1\rangle$. Therefore, we cannot easily assume one of them to be exploited first. Considering that the resource graph model is defined based on a Bayesian network, which by definition requires acyclic graphs, we cannot add bi-directional dotted lines between exploits, either.

Another related limitation is that, once exploits are considered to be partially ordered, the attack likelihood will not necessarily be the lowest when all the resources are assumed to be distinct. For example, in Figure 4, an attacker may reach condition $\langle user, 2\rangle$ through two paths, $\langle http, 0, 1\rangle \rightarrow \langle http, 1, 2\rangle$ and $\langle firewall, 0, F\rangle \rightarrow \langle http, 0, 2\rangle$. Intuitively, the attack likelihood will actually be higher if the *http* exploits in the two paths are assumed to be distinct, since now an attacker would have more choices in reaching the goal condition $\langle user, 2\rangle$. Those limitations will be addressed in following sections through a redesigned model.

[2]The dashed line and box, and the CPT table may be ignored for the time being.

B. Redesigning d_3 Metric

To address the aforementioned limitations of the original d_3 metric [14], we redesign the model of reusing exploits of the same resource type. Intuitively, what allows an attacker to more likely succeed in exploiting a previously exploited type of resources is the knowledge, skills, or exploit code he/she has obtained. Therefore, instead of directly modeling the casual relationship between reused exploits, we explicitly model such advantages of the attacker as separate events, and model their effect of increasing the likelihood of success in subsequent exploits as conditional probabilities.

More specifically, a new parent node common to exploits of the same resource type will be added to the resource graph, as demonstrated in Figure 4 using dashed lines and box. This common parent node represents the event that an attacker has the capability to exploit that type of resources. However, unlike nodes representing initial conditions, which will be treated as evidence for calculating the posterior probability of the goal condition, the event that an attacker can exploit a type of resources will not be considered observable. Adding a common parent node to exploits of the same resource type will introduce probabilistic dependence between the children nodes such that satisfying one child node will increase the likelihood of others, which models the effect of reusing exploits.

For example, in Figure 4, the dashed line box indicates a new node $\langle http\rangle$ representing the event that an attacker has the capability to exploit *http* resources. The dashed lines represent conditional probabilities that an attacker can exploit each *http* instance, and the CPT table shows an example of such conditional probability for $\langle http, 1, 2\rangle$. The marginal probability 0.08 assigned to $\langle http\rangle$ represents the likelihood that an attacker has the capability of exploiting *http* resources, and the conditional probability 0.9 assigned to $\langle http, 1, 2\rangle$ represents the likelihood for the same attacker to exploit that particular instance. The existence of such a common parent will introduce dependence between those *http* exploits, such that satisfying one will increase others' likelihood.

Formally, Definition 6 characterizes network diversity using this approach. In the definition, the second set of conditional probabilities represent the probability that an attacker is capable of exploiting each type of resources. The third and fourth sets together represent the semantics of a resource graph. Finally, the fifth set represents the conditional probability that an exploit may be executed when its pre-conditions are satisfied (including the condition that represents the corresponding resource type).

Definition 6 (d_3 Diversity): Given a resource graph $G(E \cup C, R_r \cup R_i)$, let $R' \subseteq R$ be the set of resource types

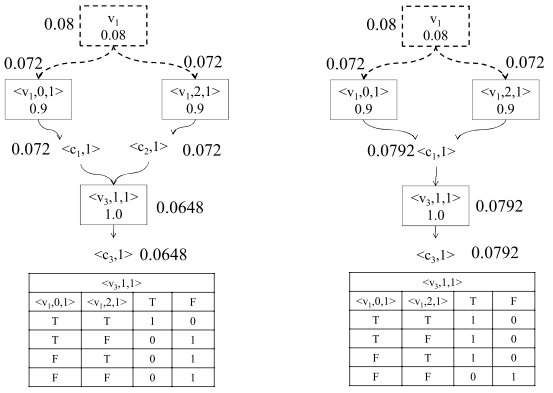


Fig. 5. Two examples of applying d_3.

each of which is shared by at least two exploits in E, and let $R_s = \{(r, \langle r, h_s, h_d\rangle) : r \in R', \langle r, h_s, h_d\rangle \in E\}$ (that is, edges from resource types to resource instances). Construct a Bayesian network $B = (G'(E \cup C \cup R', R_r \cup R_i \cup R_s), \theta)$, where G' is obtained by injecting R' and R_s into the resource graph G, and regarding each node as a discrete random variable with two states T and F, and θ is the set of parameters of the Bayesian network given as follows.

1) $P(c = T) = 1$ for all the initial conditions $c \in C_I$.
2) $P(r = T)$ are given for all the shared resource types $r \in R'$.
3) $P(e \mid \exists c_{\langle c,e\rangle \in R_r} = F) = 0$ (that is, an exploit cannot be executed until all of its pre-conditions are satisfied).
4) $P(c \mid \exists e_{\langle e,c\rangle \in R_i} = T) = 1$ (that is, a post-condition can be satisfied by any exploit alone).
5) $P(e \mid \forall c_{\langle c,e\rangle \in R_r \cup R_s} = T)$ are given for all $e \in E$ (that is, the probability of successfully executing an exploit when its pre-conditions have all been satisfied).

Given any $c_g \in C$, the network diversity d_3 is defined as $d_3 = \frac{p'}{p}$ where $p = P(c_g \mid \forall c_{c \in CI} = T)$ (that is, the conditional probability of c_g being satisfied given that all the initial conditions are true), and p' denotes the minimum possible value of p when some edges are deleted from R_s (that is, the lowest attack likelihood by assuming certain resource types are no longer shared by exploits).

Figure 5 shows two simple examples in which the first depicts a conjunction relationship between the two exploits (in the sense that both upper exploits must be executed before the lower exploit can be reached), whereas the second a disjunction relationship (any of the two upper exploits can alone lead to the lower exploit). In both cases, assuming $c_g = \langle c_3, 1\rangle$, the probability $p = P(c_g \mid \forall c_{c \in CI} = T)$ is shown in the figure. We now consider how to calculate the normalizing constant p'. For the left-hand side case, the probability $p = P(c_g \mid \forall c_{c \in CI} = T)$ would be minimized if we delete both edges from the top node (v_1) to its two children (that is, those two exploits no longer share the same resource type). It can be calculated that $p' = 0.0064$, and hence the diversity $d_3 = \frac{0.0064}{0.0648}$ in this case. The right-hand case is more interesting, since it turns out that p is already minimized because deleting edges from the top node (v_1) will only result in a higher value of p (since an attacker would have two different ways for reaching the lower exploit), which can be calculated as 0.1536. Therefore, diversity in this case is $d_3 = \frac{0.0792}{0.0792}$, that is, improving diversity will not enhance (in fact it hurts) security in this case. This example also confirms our earlier observation that assuming all resources to be distinct does not necessarily lead to the lowest attack likelihood.

The above example also leads to the observation that the normalizing constant p' may not always be straightforward to calculate since finding the case in which p is minimized essentially means we need to optimize a network's diversity for improving its security, which itself comprises an interesting future direction. Instead, we propose an approximated version of normalizing constant p' based on the following observations from the above example. In Figure 5, we can see that the right-hand side contains two attack paths leading to the goal condition $\langle c_3, 1\rangle$ (since each of the upper exploits alone is sufficient to lead to the lower exploit). We have shown previously that deleting dashed lines will only increase the probability p (of reaching the goal condition). However, we can easily see that, whether we delete the dashed lines or not, the probability p would always be minimized if there were only one path (e.g., by deleting $\langle v_1, 2, 1\rangle$ from the figure). Note that, for the left-hand side, there is already only one path since both upper exploits are required to reach the lower exploit, so p is minimized when the two upper exploits are assumed to be distinct. Intuitively, the network's security can never exceed the case in which only the shortest path (in terms of the number of steps) remains in the resource graph, with no resource being shared along the path. This intuition leads to following result.

Proposition 1: The normalizing constant p' in Definition 6 always satisfies $p' \geq p''$ where p'' is the probability $P(c_g \mid \forall c_{c \in C_I} = T)$ calculated on the shortest attack path in terms of steps (see Section III-A).

Proof (Sketch): We prove the result by mathematical induction on i, the number of steps in the shortest path. The base case $i = 1$ is trivial. For the inductive case, suppose the result holds for any resource graph with shortest path no longer than k. Given a resource graph G whose shortest path has $k+1$ steps, let the set of exploits that are directly adjacent to the goal condition c_g be E_{k+1}. Clearly, $P(c_g \mid \forall c_{c \in CI} = T) \geq P(e \mid \forall c_{c \in CI} = T)$ holds for all $e \in E_{k+1}$ since c_g can be satisfied by any exploit in E_{k+1} (and the probability of the disjunction of events cannot be smaller than the probability of any event). Without loss of generality, suppose $e_{k+1} \in E_{k+1}$ is the exploit next to c_g on the shortest path, and we have $P(c_g \mid \forall c_{c \in CI} = T) \geq P(e_{k+1} \mid \forall c_{c \in CI} = T)$. Let E_k be the set of exploits closest to e_{k+1}, $E \subseteq E_k$ be the set of exploits on the shortest path, and $e_k \in E$ be the exploit next to e_{k+1}. There cannot be any conjunctive relationships between the exploits in E and any other exploit in $E_k \setminus E$ with respect to e_{k+1}, because otherwise the shortest path would have more than $k+1$ steps, contradicting our assumption. Therefore, we have that $P(c_g \mid \forall c_{c \in CI} = T) \geq P(e_{k+1} \mid \forall c_{c \in CI} = T) \geq P(e_k \mid \forall c_{c \in CI} = T) \cdot P(e_{k+1} \mid \forall c_{\langle c,e\rangle \in R_r \cup R_s} = T)$. Then by our inductive hypothesis, we have that $P(e_k \mid \forall c_{c \in CI} = T)$ must be no less than the same probability calculated on

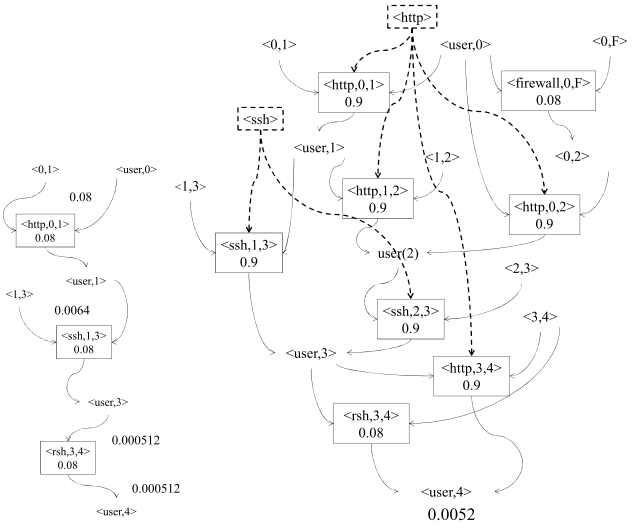


Fig. 6. Applying d_3 on the running example.

the shortest path (of length k), and hence conclude the proof. □

The above result simplifies the application of d_3 since the shortest path can be easily obtained using the algorithm mentioned in Section III-B. We apply the approach to our running example, as shown in Figure 2. Based on Table I, the first and second attack paths have the lowest number of steps. The left-hand side of Figure 6 depicts the first path. The normalizing constant can be calculated based on this path as $p' = 5.12 * 10^{-4}$. The right-hand side depicts the application of our model for reusing exploits, which adds a common parent for the same type of resources, represented using dotted lines and boxes. There are two types of resources that are reused in this resource graph, $http$ and ssh. By applying the method described above, we obtain the attack likelihood $p = 0.0052$, and therefore the network diversity can be calculated as $d_3 = \frac{p'}{p} = \frac{5.12*10^{-4}}{0.0052}$.

V. Applying the Network Diversity Metrics

The network diversity metrics we have proposed are based on abstract models of networks and attacks. How to instantiate such models for a given network is equally important. This section discusses various practical issues in applying the metrics and provides a case study on instantiating the models.

A. Guidelines for Instantiating the Network Diversity Models

To apply the proposed network diversity metrics, necessary input information needs to be collected. We describe how such inputs may be collected from a given network and discusses the practicality and scalability.

1) The d_1 Diversity Metric: To instantiate d_1, we need to collect information about

- hosts (e.g., computers, routers, switches, firewalls),
- resources (e.g., remotely accessible services), and
- similarity between resources.

Information about hosts and resources is typically already available to administrators in the form of a network map. A network scanning will assist in collecting or verifying information about active services. A close examination of host configurations (e.g., the status of services and firewall rules) may also be necessary since a network scanning may not reveal services that are currently disabled or hidden by security mechanisms (e.g., firewalls) but may be re-enabled once the security mechanisms are compromised.

Collecting and updating such information for a large network certainly demands substantial time and efforts. Automated network scanning or host-based tools exist to help simplify such tasks. Moreover, focusing on remotely accessible resources allows our model to stay relatively manageable and scalable, since most hosts typically only have a few open ports but tens or even hundreds of local applications. A challenge is to determine the similarity of different but related resources, which will be discussed in further details in Section VII.

2) The d_2-Diversity Metric: To instantiate the least attacking effort-based d_2 network diversity metric, we need to collect the following, in addition to what is already required by d_1,

- connectivity between hosts,
- security conditions either required for, or implied by, the resources (e.g., privileges, trust relationships, etc.), and
- critical assets.

The connectivity information is typically already available as part of the network map. A network scanner may help to verify such information. A close examination of host configurations (e.g., firewall rules) and application settings (e.g., authentication policies) is usually sufficient to identify the requirements for accessing a resource (pre-conditions), and an assessment of privilege levels of applications and the strength of isolation around such applications will reveal the consequences of compromising a resource (post-conditions). Critical assets can be identified based on an organization's needs and priorities.

The amount of additional information required for applying d_2 is comparable to that required for d_1, since a resource typically has a small number of pre- and post-conditions. Once such information is collected, we can construct a resource graph using existing tools for constructing traditional attack graphs due to their syntactic equivalence, and the latter is known to be practical for realistic applications [24], [25].

3) The d_3-Diversity Metric: To instantiate the probabilistic network diversity metric d_3, we need to collect the following, in addition to what is already required for d_2,

- marginal probabilities of shared resource types, and
- conditional probabilities that resources can be compromised when all the pre-conditions are satisfied.

Both groups of probabilities represent the likelihood that attackers have the capability of compromising certain resources. A different likelihood may be assigned to each resource type, if this can be estimated based on experiences or reputations (e.g., the history of past vulnerabilities found in the same or similar resource). When such an estimation is not possible or desirable (note that any assumption about attackers' capabilities may weaken security if the assumption

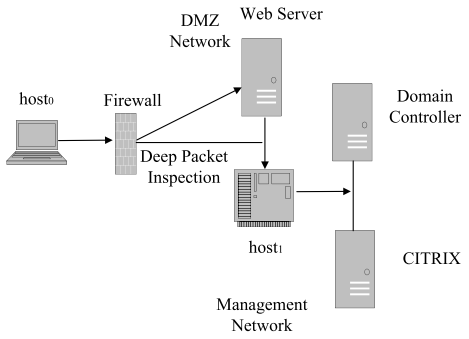


Fig. 7. The network topology of an existing work [29].

turns to be invalid), we can assign the same nominal value as follows. Since a zero day vulnerability is commonly interpreted as a vulnerability not publicly known or announced, it can be characterized using the CVSS base metrics [26], as a vulnerability with a remediation level *unavailable*, a report confidence *unconfirmed*, and a maximum overall base score (and hence produce a conservative metric value). We therefore obtain a nominal value of 0.8, converting to a probability of 0.08. For reference purpose, the lowest existing CVSS score [27] is currently 1.7, so 0.08 is reasonably low for a zero day vulnerability. Once the probabilities are determined, applying d_3 amounts to constructing Bayesian networks and making probabilistic inferences based on the networks, which can be achieved using many existing tools (e.g., we use OpenBayes [28]). Although it is a well known fact that inference using Bayesian networks is generally intractable, our simulation results have shown that the particular inference required for applying the d_3 metric can actually be achieved under reasonable computational cost [14].

B. Case Study

We present a case study to demonstrate how our models may be instantiated by following the guidelines provided in the previous section. To show the applicability of our model, the case study will be based on the network topology of an existing work [29]. Despite its relatively small scale, the network configuration mimics a typical enterprise network, e.g., with DMZ, Web server behind firewall accessible from public Internet, and a private management network protected by the same firewall but with deep packet inspection and equipped with a domain controller and CITRIX server, as shown in Figure 7. The following describes in details how we collect necessary input information for instantiating each metric, and the collected information is listed in Table II.

1) The d_1 Metric: The information we collect for instantiating d_1 includes:

- Hosts: The network topological map clearly indicates these are the hosts: Firewall, Web Server, host$_1$, Citrix, and Domain Controller.
- Resources: The network configuration description indicates the firewall used in this network is Symantec Endpoint Protection, which deploys two different rules, for the DMZ network with egress traffic filtered and for the Management network with deep content inspection. We use nmap to scan the internal network in order to collect information about open ports, applications running on the hosts and operating systems' information on the hosts, etc. For example, we determined that the public web server has open ports 80 and 43, with SQLite and Appache running on top of Ubuntu 11.04.
- Similarity between resources: We take a simplistic approach of regarding resources in this network as either identical or different so the similarity score is either 1 or 0 (this can be refined by leveraging existing tools, as discussed in Section VII).

TABLE II
COLLECTED INFORMATION

Hosts	Connectivity	Ports	Resources	Security Conditions
Firewall	Web Server, $host_1$	-	Egress traffic filtered, Deep content inspection rules	-
Web server	firewall,$host_1$	80,43	Http Services, SQLite1.2.4 , Ubuntu 11.04	user, root
$Host_1$	firewall, web server, domain controller, citrix	80,3389	File Sharing, RDP Service, Windows 7,	domain user, local administrator
Citrix	domain controller, $host_1$	80,3389	Http Services, Citrix Xen App, RDP Service	user, local administrator
Domain Controller	citrix,$host_1$	3389	RDP Service	user,domain administrator

2) The d_2 Metric: To instantiate d_2, we need to collect the following, in addition to what is already collected for d_1:

- Connectivity: the network topological map clearly provides the connectivity between hosts.
- Security conditions: we study the applications and their existing vulnerabilities in order to collect corresponding security-related pre- and post- conditions. For example, SQLiteManager version 1.2.4 runs under user privilege on Web server, which indicates a post-condition of user privilege on the host, whereas Ubuntu 11.04 has root privilege as its post-condition due to potential privilege escalation vulnerabilities (there in fact exist such vulnerabilities [30]).
- Critical assets: in this network we consider the Domain Controller as critical asset due to its special role (actual system administrators will be in a better position to designate their critical assets).

3) The d_3 Metric: To instantiate d_3, we need to collect the following, in addition to what is already collected for d_2,

- Marginal probabilities of shared resource types and conditional probabilities that resources can be compromised when all the pre-conditions are satisfied: we assign 0.08 as a nominal value for both probabilities which may certainly be refined if additional information is available to administrators (see Section V-A3 for details).

VI. SIMULATION

In this section, we study the three proposed metrics by applying them to different use cases through simulations.

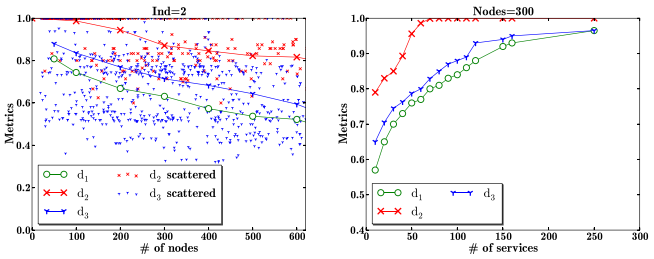


Fig. 8. Comparison of metrics (Left) and the effect of increasing diversity (Right).

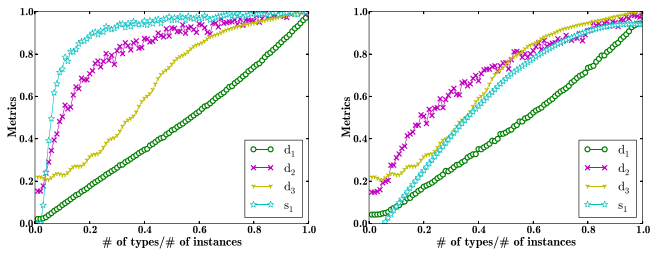


Fig. 9. Worm propagation (Left 10% initially satisfied exploits, Right 50% initially satisfied exploits).

All simulation results are collected using a computer equipped with a 3.0 GHz CPU and 8GB RAM in the Python environment under Ubuntu 12.04 LTS. The Bayesian network-based metric is implemented using OpenBayes [28]. To generate a large number of resource graphs for simulations, we first construct a small number of seed graphs based on real 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., we vary the number of pre-conditions of each exploit within a small range since real world exploits usually have a small number of pre-conditions).

We apply the three network diversity metrics to different use cases, as presented in section II-A. Our objective is to evaluate the three metrics through numerical results and to examine those results together with statistically expected results represented by different attack scenarios.

The first two simulations compare the results of all three metrics to examine their different trends as graph sizes increase and as diversity increases. First of all, to convert the Bayesian network-based metric d_3 to a comparable scale of the other two, we use $\frac{\log_{0.08}(p')}{\log_{0.08}(p)}$ (i.e., the ratio based on equivalent numbers of zero day exploits) instead of d_3. In the left-hand side of Figure 8, the scatter points marked with X in red are the individual values of d_2. The blue points marked with Y are the values of d_3 (converted as above). Also shown are their average values, and the average value of the effective richness-based metric d_1. The right figure shows the average value of the three metrics in increasing number of distinct resources for resource graphs of a fixed size.

Results and Implications: Both simulations show that, while all three metrics follow a similar trend (in the left figure, diversity will decrease in larger graphs since there will be more duplicated resources) and capture the same effect of increasing diversity (in the right figure), the Bayesian network-based metric d_3 somehow reflects an intermediate result between the two other extremes (d_1 can be considered as the average over all resources, whereas d_2 only depends on the shortest path). These results show that applying all three metrics may yield consistent results and motivates us to compare them through further simulations.

Next we examine the metric results under different use cases, as described in Section II-A. The first use case considers worms characterized as follows. First, each worm can only exploit a small number of vulnerabilities. In our implementation, we randomly choose one to three resource types as the capability of each worm. Second, the goal of a worm is infecting as many hosts as possible, does not need specific targets. Although some worms or bots may in reality have a target, it is usually still necessary for them to first compromise a large number of machines before the target can be reached (e.g., Stuxnet [1]). In Figure 9, the X-axis is the ratio of the number of resource types to the number of resource instances, which roughly represents the level of diversity in terms of richness (it can be observed that d_1 is close to a straight line). Y-axis shows the results of the three metrics as well as the ratio of hosts that are not infected by the simulated worms. The four lines represent the three metrics (marked with d_1, d_2, and d_3) and the ratio of hosts uninfected by simulated worms (marked with S_1). The left and right figures correspond to different percentages of first-level exploits (the exploits that only have initial conditions as their pre-conditions) among all exploits, which roughly depicts how well the network is safeguarded (e.g., 50% means a more vulnerable network than 10% since initially attackers can reach half, or five times more, exploits). For each configuration, we repeat 500 times to obtain the average result of simulated worms.

Results and Implications: In this simulation, we can make the following observations. First of all, all three metrics still exhibit similar trends and relationships as discussed above. The left figure shows that, when the network is better safeguarded (with only 10% of exploits initially reachable), the effect of increasing diversity on simulated worms shows a closer relationship with the d_2 metric than the other two, both of which indicate that increasing diversity can significantly increase the percentage of hosts uninfected by worms. In comparison, the right figure shows a less promising result where the three metrics and the percentage of uninfected hosts all tend to follow a similar trend. Intuitively, in well guarded networks, many hosts cannot be reached until the worms have infected other adjacent hosts, so increasing diversity can more effectively mitigate worm propagation. In less guarded networks where half of the exploits may be reached initially, the effect of diversity on worms is almost proportional to the richness of resources (d_1), and all three metrics tend to yield similar results.

The second use case is targeted attacks (Section II-A). We simulate attackers with different capabilities (sets of resources they can compromise) and the level of such capabilities (that is, the number of resources they can compromise) follows the Gamma distribution [31]. Similarly, we also repeat each experiment 500 times and we examine two different cases corresponding to different percentages of first-level exploits.

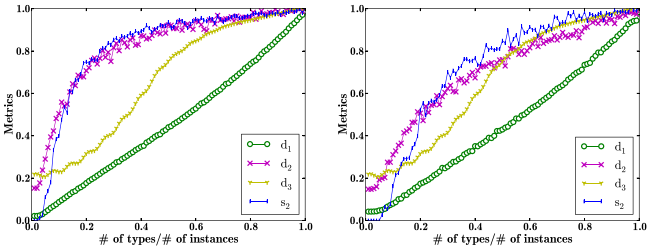


Fig. 10. Targeted attack (Left 10% initially satisfied vulnerabilities, Right 50% initially satisfied vulnerabilities).

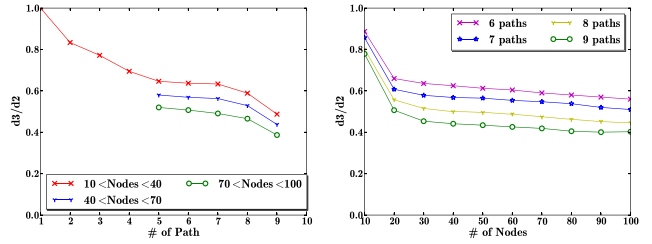


Fig. 11. d_3/d_2 in the Number of Paths (Left) and Nodes (Right).

In Figure 10, S_2 is the result of simulated attacker, which means the percentages of attackers who cannot reach the randomly selected goal condition.

Results and Implications: From the results we can observe similar results as with the simulated worms. Specifically, increasing diversity can more effectively mitigate the damage caused by simulated attackers for well guarded networks (the left figure) than for less guarded networks (the right figure). Also, in the left figure, the simulated attackers' results are closer to that of d_2 than the other two metrics, whereas it is closer to both d_2 and d_3 in the right figure. In addition, by comparing the results in Figure 10 (targeted attack) to that in Figure 9 (worm), we can see that the same level of diversity can more effectively mitigate worm than it can do to simulated attackers. This can be explained by the fact that a worm is assumed to have much less capability (set of resources it can compromise) than a simulated attacker.

The next set of simulations examines the relationships between d_2 and d_3 in more details. Those two metrics both take into account the causal relationships between resources, but they focus on slightly different perspectives (the least and average efforts needed to compromise a network). At the same time, the two metrics are closely related. In previous simulations we already see that the values of d_3 are almost always smaller than d_2. The two metrics may also converge to similar values in certain cases (e.g., in an extreme case of only one path in the resource graph, d_2 and d_3 will yield identical value). To see how those two metrics relate to each other, we simulate resource graphs of various sizes and shapes and the average results (d_3/d_2) of 500 simulations are shown in Figure 11.

Results and Implications: In Figure 11, the left figure shows that, under fixed sizes of resource graphs, the difference between d_2 and d_3 increases (a ratio of 1 means they have identical values) with the number of paths in the resource graphs. This is expected since d_2 only represents diversity in terms of one specific path while d_3 accounts all paths so with the number of paths increasing the ratios becomes smaller (meaning larger differences). The right figure shows that the differences increase (the ratio decreases) as graph sizes increase (all resource graphs have between 6 to 9 paths). This can be explained by the fact that the difference between different paths will become more significant as the size of graphs increases and with the number of paths fixed, and therefore the difference between the two metrics slowly increases (the ratio decreases). Both simulations imply that, although the choice of metrics mostly depends on the desired perspective, both metrics should be considered especially for larger and less well guarded networks (meaning more attack paths are present) since their values may be vastly different.

We now study the third use case, the Moving Target Defense (MTD). The MTD approach attempts to achieve better security by varying in time the configurations of networks, in which diversity plays a critical role. Our goal here is to study the effect of varying diversity on the effectiveness of MTD, and also on evaluating our metric when applied to MTD. To the best of our knowledge, this is among the first efforts on studying MTD using simulations (another similar effort by Zhuang et al. [32] also employs simulation results, but our goal is to evaluate the proposed network diversity metric under MTD applications, which is different from their study). Our simulations are based on following assumptions.

- We assume attackers do not initially have full details about the network but may gradually learn about such details as the configuration is changed over time. Specifically, attackers can learn about the change of a configuration (e.g., either through the failure of their attacks, or by observing special features of a configuration).
- We assume attackers' capabilities, which are sets of resources they can compromise, follow gamma distribution, and each resource has an associated time window indicating the only duration in which the resource may be compromised by attackers who have corresponding capabilities. Moreover, having the capabilities does not mean the attacker can immediately compromise the resource, since he/she may still need certain amount of time to actually implement the attack on specific instances of the resource. Therefore, in our simulations, we assign each resource an attack window, and only a resource whose duration of appearing inside a configuration is longer than the corresponding attack window may be compromised by attackers who have the capability.
- We assume the MTD approach employs dynamically changing network configurations. Specifically, the network topology remains the same but resources of each host may change. Also, we assume a fixed frequency of changes in our simulations (e.g., in Figure 12, the left figure shows the frequency of configuration change is 1 configuration/per day). We leave other ways for changing network configurations, such as using a varying frequency, as future work.

In Figure 12, the left figure shows the average success rates of attackers after 40 days of exposure to the network in the

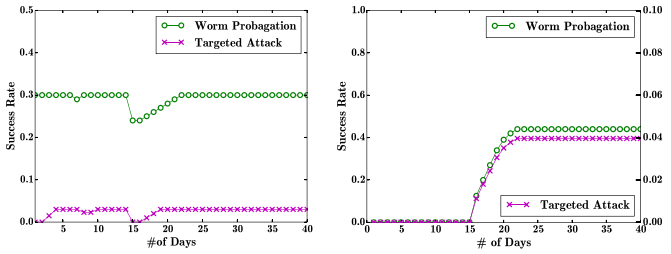


Fig. 12. Success rate of attacks in frequency of changes (Left) and in time (Right).

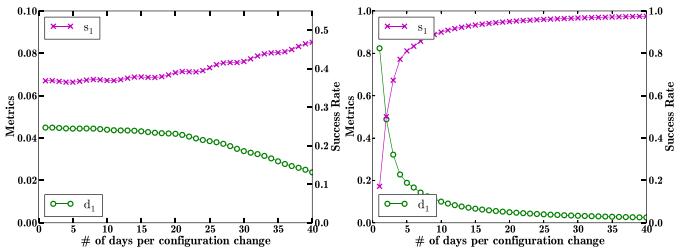


Fig. 13. d_1 in Frequency of changes, under less resource types (Left) and more resource types (Right).

number of days before a configuration changes (e.g., 5 day means there is one configuration change per five days). The right figure shows the attack success rates (the left Y-axis represents the success rate of worms and the right for targeted attacks) in the number of days since the network is exposed, with the frequency of changes set at one configuration change per day.

Results and Implications: From the left figure, we can see that a more frequent change of network configurations does not necessarily imply better security (lower success ratio), since too fast or too slow changes can both increase the exposure of a resource and hence increase an attacker's chance of compromising that resource. In fact, the left figure indicates no clear trend in the success rate as the number of days for a change increases. The small drops in both lines indicate that the lowest success rates coincident with the average size of attack windows (in this case we assume 10 or 15 days are required to compromise a resource), which may not be meaningful in reality since different resources may have different attack windows. From the right figure, we can see that, before 15 days, there is zero success rate for both worm propagation and targeted attack, due to the assumed 10 to 15 days of attack windows. After 15 days, there is a jump in both lines, which means, with accumulated efforts, both worms and attackers will be able to compromise more and more resources, and the success rates do not change much after about 20 days since now they will depend more on the capability of worms (attackers).

In Figure 13, we only apply the d_1 metric to MTD since all three metrics will show similar trends as discussed above. In the left figure, we can see that if the set of resources types remains at a relatively small size (e.g., $\frac{\#of\ Resources}{\#of\ Days} < 20\%$), then our d_1 metric stays almost flat when the frequency of configuration changes is relatively high, and it then drops more dramatically when the frequency is lower (around

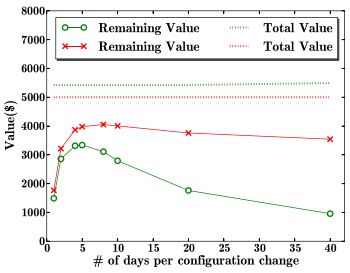


Fig. 14. Total cost in frequency of changes.

one change per 20 days). This indicates that, with limited number of resource types, a higher frequency of configuration changes does not provide much security gain, unless if the frequency is too low. The left figure also indicates that, when the number of days per change increases over 20, diversity drops while success rate increases, which means our diversity metric effectively capture the effectiveness of MTD. The right figure shows similar results under larger set of resources ($\frac{\#of\ Resources}{\#of\ Days} > 80\%$). The successful rate and d_1 shows corresponding (reversed) trends. However, now with a large enough set of resources to choose from, less frequent changes in configurations mean lower diversity and hence less security.

In this last simulation, we study the relationship between cost and security in MTD, as shown in Figure 14. For worm propagation, we assign monetary values to all hosts, with critical assets (goal conditions) having higher values, whereas for targeted attack, we only assign a value to critical assets. The red and green dashed lines on top of the figure shows the total value for each scenario. Each time when worms or attackers compromise a resource, its assigned value is considered lost. Such lost value is the first part of overall cost. The other part is the cost of changing configurations in MTD (e.g., administrative cost of purchasing new software or performance cost of delaying a client's request).

Results and Implications: Figure 14 depicts the total cost (lost value due to compromised resources plus cost of configuration changes) in the number of days for a change of configuration. The results show that the overall cost will first decrease and then increase. The optimal setting of frequency is about one configuration change per 5 days, which best balances the cost of changing new configuration with lost values of compromised resources. Note that, according to our discussions above, the optimal frequency will also depend on the number of available resources.

VII. Discussion

The similarity between resources is an important input to our metric models. In Section V-B, we have taken a simplistic approach of regarding resources as either identical or completely different (the similarity score is either 1 or 0). Although such an approach may be acceptable in many cases, it certainly needs to be refined considering the fact that slight differences usually exist among different versions of the same software, whereas different software may share common code or libraries. Measuring such differences or similarity between resources will lead to more accurate results for the

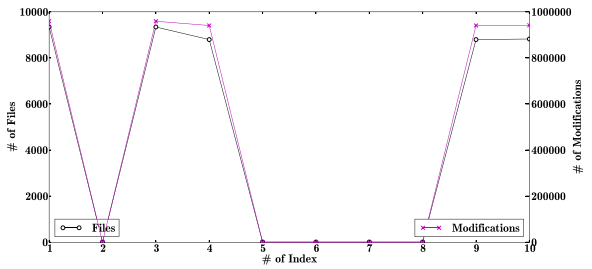


Fig. 15. Differences at file and modification levels between different versions of chrome.

diversity metrics. This section demonstrates such a need and discusses potential solutions.

A. A Case Study

To demonstrate the need for studying software similarity, we conduct a case study on different versions of Chrome, and show how such differences when taken into account may affect the diversity metric results. Specifically, we examine 11 consecutive versions of the Chrome browser between versions 42.0.2283.5 (published Jan 22 06:24:31 2015) and 41.0.2272.16 (published Jan 21 02:44:41 2015) all of which are published along two development branches 41 and 42. The versions are labeled with index numbers between 0 to 10, from the latest to the oldest, in Figure 15. We use the latest version 0 (42.0.2283.5) as the baseline for comparison.

It might be expected that not much difference should exist between those versions with such small differences in version numbers and publish time. However, our study shows this is not really the case. Taking versions 0 (42.0.2283.5) and 1 (41.0.2272.28) as an example, while the total number of source files is quite similar (75136 and 73596, respectively), there are differences in totally 9338 files (about 12%) between the two versions. Moreover, such differences include 767,987 insertions and 190,943 deletions, which accounts for about 13% of lines (the total number of lines in those two versions are 14,750,264 and 15,330,677, respectively). Those numbers clearly indicate a significant difference between the two versions even though their publish time are only a few minutes apart.

In Figure 15, the two lines depict the number of differences in terms of files and modifications, respectively, for the ten versions. Clearly, there is a similar trend at those two different levels of differences, with smaller differences among versions in the same branch (e.g., 42) and more significant differences between different branches (the numbers can reach almost 10,000 at file level and 1,000,000 at modification level). Although the version numbers may seem to provide useful information in this particular case, such information about the relationships between multiple versions (e.g., development branches) may not always be provided by a software vendor. Therefore, it is not a reliable approach to rely on either the version numbers or publish time to determine the similarity between multiple versions.

B. Potential Solutions

In addition to slight differences between multiple versions of the same software, there may also exist similarity between completely different software. Most of today's complex software are developed under established software engineering principles which encourage modularity, software reuse, the use of program generators, iterative development, and open-source packages. As a result, many software may share common code blocks or employ common libraries, both of which may result in significant similarity between those software. For example, Chrome and Opera have both been using Blink as their rendering engine since 2013 so both include common code blocks. A well known example of sharing common library functions in different software is the recent Heartbleed bug in which many popular Web servers, including Apache HTTP Server, IBM HTTP Server, and Nginx, all employ the common openssl library to establish SSL connections. To measure such similarity between different software, we believe existing efforts, such as clone detection at both source code and binary levels, should be leveraged and extended to develop practical solutions. Although this is not the main focus of this paper, we provide a brief overview of existing approaches which we believe are promising in this regard.

At source code level, there exists a rich literature on clone detection, which attempts to match code fragments through both syntax and semantic features. For example, a text-based approach extracts signatures of the lines and then matches software based on substrings [33]. Such a technique provides basic matching results although it has many limitations, e.g., it cannot handle identifier renaming, since it does not transform source code into intermediate formats. A token-based approach parses the source code into a list of tokens [34] so that it can handle renaming, although it has its own limitations, e.g., it cannot deal with replacement of control flows. In a tree-based approach [35], an abstract syntax tree is generated from the source code for matching. Such technique provides more semantic features but it ignores data flows and therefore cannot deal with reordering statements. Apart from those matching-based approaches, a similarity distance-based approach [36] calculates a distance between two code segments and then compares to a given threshold. There exist many other approaches in the literature, all of which may be leveraged to identify and quantify similarity between open source software.

As to closed source software, identifying shared code or library functions is more challenging. Nonetheless, there are many existing efforts on assembly-level clone detection and library identification. The text-based approach regards the executable part of a binary as a sequence of bytes or lines of assembly and compares them to find identical sequences [37]. The token-based approach relies on feature vectors consisted of opcodes and operands and employs metrics to provide function-level clone detection [38]. The structural-based approach maps the code back to execution schema and compares their structural features [39]. Our recent work combines several existing concepts from classic program analysis, including control flow graph, register flow graph, and function call graph, to capture semantic similarity between two binaries [40]. Finally, the binary search engine provides an easy way for locating shared libraries inside a software binary [41]. Although more challenging

than it is for open source software, we believe developing practical tools by leveraging such existing efforts to identify and estimate similarity for closed source software is still possible.

Finally, variations of software may also be caused by configuration differences (e.g., different settings of Sophos antivirus software), additional security hardening measures (e.g., SELinux and Grsecurity), add-ons and plugins, etc., which may sometimes offer even more substantial impact than different versions of a software. Taking into account such factors in measuring software similarity can be a real challenge and the only tangible solution may still be relying on administrators' manual inspection and estimation.

VIII. Related Work

The research on security metrics has attracted much attention lately. Unlike existing work which aim to measure the amount of network security [42], [43], this paper focuses on diversity as one particular property of networks which may affect security. Nonetheless, our work borrows from the popular software security metric, attack surface [44], the general idea of focusing on interfaces (remotely accessible resources) rather than internal details (e.g., local applications). Our least attacking effort-based diversity metric is derived from the k-zero day safety metric [45], [46], and our probabilistic diversity metric is based on the attack likelihood metric [23], [47]. Another notable work evaluates security metrics against real attacks in a controlled environment [48], which provides a future direction to better evaluate our work. One limitation of our work lies in the high complexity of analyzing a resource graph; high level models of resource dependencies [49] may provide coarser but more efficient solutions to modeling diversity.

The idea of using design diversity for fault tolerance has been investigated for a long time. The N-version programming approach generates $N \geq 2$ functionally equivalent programs and compares their results to determine a faulty version [50], with metrics defined for measuring the diversity of software and faults [51]. The main limitation of design diversity lies in the high complexity of creating different versions, which may not justify the benefit [52]. The use of design diversity as a security mechanism has also attracted much attention [53]. The general principles of design diversity is shown to be applicable to security as well in [2]. The N-Variant system extends the idea of N-version programming to detect intrusions [3], and the concept of behavioral distance takes it beyond output voting [4]. Different randomization techniques have been used to automatically generate diversity [7]–[9], [54]. Those techniques can certainly be applied to improve the network diversity measured using our metric model. In another word, our metric provides a quantitative method for applying and evaluating such methods.

In addition to design diversity and generated diversity, recent work employ opportunistic diversity which already exists among different software systems. The practicality of employing OS diversity for intrusion tolerance is evaluated and the feasibility of using opportunistic diversity already existing between different OSes to tolerate intrusions is demonstrated in [6]. Diversity has also been applied to intrusion tolerant systems which usually implement some kinds of Byzantine Fault Tolerant (BFT) replication as fault tolerance solutions [5]. A generic architecture for implementing intrusion-tolerant Web servers based on redundancy and diversification principles is introduced in [55]. Components-off-the-shelf (COTS) diversity is employed to provide an implicit reference model, instead of the explicit model usually required, for anomaly detection in Web servers [56]. Diversity could play an important role in addressing various security issues in cloud computing [10], such as using diverse authorities for efficient decryption and revocation in cloud storage [57] and using diverse access policies for increasing the security of cloud data [58].

IX. Conclusion

In this paper, we have taken a first step towards formally modeling network diversity as a security metric for evaluating networks' robustness against zero day attacks. We first devised an effective richness-based metric based on its counterpart in ecology. We then proposed a least attacking effort-based metric to address causal relationships between resources and a probabilistic metric to reflect the average attacking effort. We provided guidelines for instantiating the proposed metrics and discussed how software diversity may be estimated. Finally, we evaluated our algorithms and metrics through simulations. Our study has shown that an intuitive notion of diversity could easily cause misleading results, and the proposed formal models provided better understanding of the effect of diversity on network security.

The main limitations of this work and corresponding future directions are as follows.

- Evaluating the proposed metrics in real production networks will provide a much stronger justification of its effectiveness.
- Although we have applied several existing biodiversity metrics, we believe more lessons could potentially be borrowed from this rich literature for improving network security.
- The redesigned probabilistic model introduced in Section IV-A can be further refined to better reflect attackers' increasing capability in repetitive exploits of similar vulnerabilities.
- Obtaining various inputs for instantiating the proposed metrics can be challenging in practice. In addition to the guidelines and case study presented in Section V, our future work will develop practical tools for gathering the inputs, e.g., estimated measures of software diversity.
- Finally, the lack of support for modeling initial exploits of client-side applications, insider attacks and user mistakes (e.g,. phishing) is a limitation of the current model and comprises an interesting future direction.

Acknowledgements

The authors thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] N. Falliere, L. O. Murchu, and E. Chien, "W32.stuxnet dossier," Symantec Security Response, Tech. Rep., 2011.

[2] B. Littlewood and L. Strigini, "Redundancy and diversity in security," in *Proc. Comput. Secur. (ESORICS)*, 2004, pp. 423–438.

[3] B. Cox *et al.*, "N-variant systems: A secretless framework for security through diversity," in *Proc. 15th Conf. USENIX Secur. Symp.*, 2006, Art. ID 9.

[4] D. Gao, M. Reiter, and D. Song, "Behavioral distance measurement using hidden Markov models," in *Recent Advances in Intrusion Detection*. Berlin, Germany: Springer-Verlag, 2006, pp. 19–40.

[5] B.-G. Chun, P. Maniatis, and S. Shenker, "Diverse replication for single-machine Byzantine-fault tolerance," in *Proc. USENIX Annu. Tech. Conf.*, 2008, pp. 287–292.

[6] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "OS diversity for intrusion tolerance: Myth or reality?" in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2011, pp. 383–394.

[7] S. Bhatkar, D. C. DuVarney, and R. Sekar, "Address obfuscation: An efficient approach to combat a broad range of memory error exploits," in *Proc. 12th USENIX Secur. Symp.*, Washington, DC, USA, 2003, pp. 105–120.

[8] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, 2003, pp. 272–280.

[9] S. Bhatkar and R. Sekar, "Data space randomization," in *Proc. 5th Int. Conf. Detection Intrusions Malware, Vulnerability Assessment (DIMVA)*, 2008, pp. 1–22.

[10] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[11] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, 1st ed. New York, NY, USA: Springer-Verlag, 2011.

[12] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks: A software diversity approach," in *Proc. 9th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2008, pp. 149–158.

[13] J. Caballero, T. Kampouris, D. Song, and J. Wang, "Would diversity really increase the robustness of the routing infrastructure against software defects?" in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2008.

[14] 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 *Proc. ESORICS*, 2014, pp. 494–511.

[15] C. S. Elton, *The Ecology of Invasions by Animals and Plants*. Chicago, IL, USA: Univ. Chicago Press, 1958.

[16] E. C. Pielou, *Ecological Diversity*. New York, NY, USA: Wiley, 1975.

[17] M. O. Hill, "Diversity and evenness: A unifying notation and its consequences," *Ecology*, vol. 54, no. 2, pp. 427–432, 1973.

[18] T. Leinster and C. A. Cobbold, "Measuring diversity: The importance of species similarity," *Ecology*, vol. 93, no. 3, pp. 477–489, 2012.

[19] K. S. McCann, "The diversity-stability debate," *Nature*, vol. 405, no. 6783, pp. 228–233, 2000.

[20] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proc. IEEE Symp. Secur. Privacy*, May 2002, pp. 273–284.

[21] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proc. ACM CCS*, 2002, pp. 217–224.

[22] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *Proc. DSN*, 2012, pp. 1–12.

[23] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic Bayesian network," in *Proc. 4th ACM QoP*, 2008, pp. 23–30.

[24] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, 2006, pp. 336–345.

[25] 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. Boston, MA, USA: Kluwer, 2003.

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

[27] (May 9, 2008). *National Vulnerability Database*. [Online]. Available: http://www.nvd.org

[28] K. Gaitanis and E. Cohen. (2013). *OpenBayes 0.1.0*. [Online]. Available: https://pypi.python.org/pypi/OpenBayes

[29] (Sep. 2015). *Penetration Testing Virtual Labs*. [Online]. Available: https://www.offensive-security.com/offensive-security-solutions/virtual-penetration-testing-labs/

[30] (Sep. 2015). *CVE for Ubuntu 11.04*. [Online]. Available: http://www.cvedetails.com/vulnerability-list/vendor_id-4781/product_id-20550/version_id-104819/Canonical-Ubuntu-Linux-11.04.html

[31] M. A. McQueen, W. F. Boyer, M. A. Flynn, and G. A. Beitel, "Time-to-compromise model for cyber risk reduction estimation," in *Quality of Protection*. New York, NY, USA: Springer-Verlag, 2006, pp. 49–64.

[32] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal, "Simulation-based approaches to studying effectiveness of moving-target network defense," in *Proc. Nat. Symp. Moving Target Res.*, 2012.

[33] J. H. Johnson, "Identifying redundancy in source code using fingerprints," in *Proc. Conf. Centre Adv. Stud. Collaborative Res., Softw. Eng.*, vol. 1. 1993, pp. 171–183.

[34] H. A. Basit, S. Jarzabek, S. J. Puglisi, W. F. Smyth, and A. Turpin, "Efficient token based clone detection with flexible tokenization," in *Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, 2007, pp. 513–516.

[35] W. S. Evans, C. W. Fraser, and F. Ma, "Clone detection via structural abstraction," *Softw. Quality J.*, vol. 17, no. 4, pp. 309–330, 2009.

[36] R. Brixtel, M. Fontaine, B. Lesner, C. Bazin, and R. Robbes, "Language-independent clone detection applied to plagiarism detection," in *Proc. 10th IEEE Working Conf. Sour. Code Anal. Manipulation (SCAM)*, 2010, pp. 77–86.

[37] J. Jang, D. Brumley, and S. Venkataraman, "BitShred: Fast, scalable malware triage," CyLab, Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CyLab-10-022, 2010.

[38] A. Sæbjørnsen, J. Willcock, T. Panas, D. Quinlan, and Z. Su, "Detecting code clones in binary executables," in *Proc. 8th Int. Symp. Softw. Test. Anal.*, 2009, pp. 117–128.

[39] T. Dullien, E. Carrera, S. M. Eppler, and S. Porst, "Automated attacker correlation for malicious code," in *Proc. NATO RTO Inf. Assurance Cyber Defence*, Tallinn, Estonia, Dec. 2010.

[40] S. Alrabaee, P. Shirani, L. Wang, and M. Debbabi, "*SIGMA*: A semantic integrated graph matching approach for identifying reused functions in binary code," *Digital Investigation*, vol. 12, pp. S61–S71, Mar. 2015.

[41] W. M. Khoo, A. Mycroft, and R. Anderson, "Rendezvous: A search engine for binary code," in *Proc. 10th Working Conf. Mining Softw. Repositories (MSR)*, 2013, pp. 329–338.

[42] N. Idika and B. Bhargava, "Extending attack graph-based security metrics and aggregating their application," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 1, pp. 75–85, Jan. 2012.

[43] L. Wang, A. Singhal, and S. Jajodia, "Toward measuring network security using attack graphs," in *Proc. 3rd ACM QoP*, 2007, pp. 49–54.

[44] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 371–386, May 2011.

[45] L. Wang, S. Jajodia, A. Singhal, and S. Noel, "*k*-zero day safety: Measuring the security risk of networks against unknown attacks," in *Proc. 15th Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2010, pp. 573–587.

[46] 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," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 1, pp. 30–44, Jan./Feb. 2013.

[47] L. Wang, A. Singhal, and S. Jajodia, "Measuring the overall security of network configurations using attack graphs," in *Proc. 21th IFIP*, 2007, pp. 98–112.

[48] H. Holm, M. Ekstedt, and D. Andersson, "Empirical analysis of system-level vulnerability metrics through actual attacks," *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 6, pp. 825–837, Nov./Dec. 2012.

[49] N. Kheir, N. Cuppens-Boulahia, F. Cuppens, and H. Debar, "A service dependency model for cost-sensitive intrusion response," in *Proc. ESORICS*, 2010, pp. 626–642.

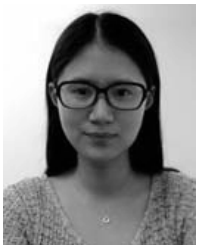
[50] A. Avizienis and L. Chen, "On the implementation of N-version programming for software fault tolerance during execution," in *Proc. IEEE COMPSAC*, Nov. 1977, pp. 149–155.

[51] S. Mitra, N. R. Saxena, and E. J. McCluskey, "A design diversity metric and analysis of redundant systems," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 498–510, May 2002.

[52] B. Littlewood, P. Popov, and L. Strigini, "Modeling software design diversity: A review," *ACM Comput. Surv.*, vol. 33, no. 2, pp. 177–208, Jun. 2001.

[53] R. A. Maxion, "Use of diversity as a defense mechanism," in *Proc. Workshop New Secur. Paradigms (NSPW)*, 2005, pp. 21–22.

[54] *PaX Address Space Layout Randomization*. [Online]. Available: http://pax.grsecurity.net/, accessed Jun. 2015.
[55] A. Saïdane, V. Nicomette, and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for Web servers," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 1, pp. 45–58, Jan./Mar. 2009.
[56] E. Totel, F. Majorczyk, and L. Mé, "COTS diversity based intrusion detection and application to Web servers," in *Proc. RAID*, 2005, pp. 43–62.
[57] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.
[58] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 2013–2021.



Mengyuan Zhang received the B.E. and M.E. degrees from the Nanjing University of Posts and Telecommunications in China. She is currently pursuing the Ph.D. degree with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada. Her research interests include network security, security metrics, and attack surface.

Lingyu Wang received the Ph.D. degree in information technology from George Mason University. He is currently an Associate Professor with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada. His research interests include data privacy, network security, security metrics, cloud computing security, and malware analysis. He has authored about 100 refereed conference and journal articles on security and privacy.

Sushil Jajodia (F'13) received the Ph.D. degree from the University of Oregon, Eugene. He is currently a University Professor, a BDM International Professor, and the Founding Director of the Center for Secure Information Systems with the Volgenau School of Engineering, George Mason University, Fairfax, VA. He is also the Founding Site Director of the NSF I/UCRC Center for Configuration Analytics and Automation at Mason. He has authored or coauthored seven books, edited 44 books and conference proceedings, and has authored over 450 technical papers in the refereed journals and conference proceedings. He holds 18 patents. His h-index is 89 and Erdos number is 2. His research interests include security, privacy, databases, and distributed systems. His current research sponsors are the Army Research Office, the Office of Naval Research, the National Security Agency, the National Science Foundation, the National Institute of Standards and Technology, Northrop Grumman Corporation, and The MITRE Corporation. He received the 1996 IFIP TC 11 Kristian Beckman Award, the 2000 Volgenau School of Engineering Outstanding Research Faculty Award, the 2008 ACM SIGSAC Outstanding Contributions Award, the 2011 IFIP WG 11.3 Outstanding Research Contributions Award, and the 2015 ESORICS Outstanding Research Award. He was recognized for the most accepted papers at the 30th anniversary of the IEEE Symposium on Security and Privacy. He has supervised 27 doctoral dissertations. Nine of these graduates hold tenured positions at U.S. universities, four are NSF CAREER awardees, and one is a DoE Young Investigator Awardee. Two additional students are tenured at foreign universities. He has served in different capacities for various journals and conferences. He is the Founding Consulting Editor of the *International Series on Advances in Information Security* (Springer) and *Computer Science* (SpringerBriefs). He was the Founding Editor-in-Chief of the *Journal of Computer Security* (1992–2010) and the Editor of the *ACM Transactions on Information and Systems Security* (1999–2006), the *IET Information Security* (2007–2014), the *International Journal of Cooperative Information Systems* (199–2011), the IEEE CONCURRENCY (1999–2000), and the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (1989–1991).

Anoop Singhal (SM'13) received the Ph.D. degree in computer science from The Ohio State University, Columbus, OH, USA. He is currently a Senior Computer Scientist with the Computer Security Division, National Institute of Standards and Technology, Gaithersburg, MD, USA. His research interests are in network security, network forensics, cloud computing security, and data mining systems. He is a member of the Association for Computing Machinery, and he has coauthored over 50 technical papers in leading conferences and journals. He has also coedited a book on Secure Cloud Computing.

Massimiliano Albanese received the Ph.D. degree in computer science and engineering in 2005 from the University of Naples Federico II, and joined George Mason University in 2011 after serving as a Postdoctoral Researcher with the University of Maryland. He is currently an Assistant Professor with the Department of Information Sciences and Technology, George Mason University, where he is also serving as the Associate Director of the Center for Secure Information Systems, and the Codirector of the Laboratory for IT Entrepreneurship. His research interests are in the area of Information and Network Security, with a particular emphasis on modeling and detection of cyber attacks, cyber situational awareness, network hardening and optimal defenses, moving target defense and adaptive cyber defense, and cyber-physical systems. He has participated in sponsored research projects totaling $7.7M. He holds a U.S. patent and has coauthored a book and over 50 papers in peer-reviewed journals and conference proceedings. He is one of the only three recipients of the 2014 Mason Emerging Researcher/Scholar/Creator Award, one of the most prestigious honors at Mason.