Extending Attack Graph-Based Security Metrics and Aggregating Their Application

Nwokedi Idika and Bharat Bhargava, *Fellow*, IEEE

Abstract—The attack graph is an abstraction that reveals the ways an attacker can leverage vulnerabilities in a network to violate a security policy. When used with attack graph-based security metrics, the attack graph may be used to quantitatively assess security-relevant aspects of a network. The Shortest Path metric, the Number of Paths metric, and the Mean of Path Lengths metric are three attack graph-based security metrics that can extract security-relevant information. However, one's usage of these metrics can lead to misleading results. The Shortest Path metric and the Mean of Path Lengths metric fail to adequately account for the number of ways an attacker may violate a security policy. The Number of Paths metric fails to adequately account for the attack effort associated with the attack paths. To overcome these shortcomings, we propose a complimentary suite of attack graph-based security metrics and specify an algorithm for combining the usage of these metrics. We present simulated results that suggest that our approach reaches a conclusion about which of two attack graphs correspond to a network that is most secure in many instances.

Index Terms—Network-level security and protection, measurement, measurement techniques.

✦

1 INTRODUCTION

A N enterprise security goal is to remove all network and host vulnerabilities. The accomplishment of this goal requires a guarantee that all vulnerabilities have been identified. Such a requirement in practice is infeasible. Humans manage networks and are prone to miss latent vulnerabilities. Even when vulnerabilities are apparent and have been identified, there may be no known viable solution to deal with the vulnerabilities. For instance, if an organization leverages Commercial-Off-the-Shelf (COTS) software to operate its network, the organization is exposing itself to any vulnerabilities the software possesses. Issues such as slow patch release times and unstable released patches may cause an organization to operate its network with known vulnerabilities.

Attacks that use existing network vulnerabilities that successfully violate a security policy, may be done with a single attack action or a series of attack actions. A series of attack actions is sometimes referred to as a chained exploit. Chained exploits leverage the interdependencies that exist among vulnerabilities to violate a network's security policy. For instance, an attacker may leverage the vulnerabilities existing at an organization's mail server and end user desktops to violate security policies. The attacker could send a malicious email to an organization with a PDF attachment that when executed by Adobe Reader deploys a Trojan Horse on the host. The Anti-Virus (AV) scanner at the mail server will fail to detect the malicious PDF because the attacker leverages a vulnerability in the AV scanner that causes the scanner to execute without scanning the attachment. Because the desktop AV scanners on end user desktops suffer from the same vulnerability, the Trojan Horse, once deployed, is not detected by the desktop AV scanner. The vulnerabilities existing in Adobe Reader and the AV scanners on the mail server and end user desktops made the above-chained exploit possible. The set of all chained exploits that violate a security policy, or a set of security policies, can be captured by an attack graph. Security-relevant information is extracted from the attack graph with attack graph analyses.

The attack graph analyses of interest in this work are those that produce security metrics. We use the Systems Security Engineering Capability Maturity Model (SSE-CMM) [1] definition of a security metric. Succinctly, a security metric (or a combination of security metrics) is a quantitative measure of how much of an identifiable security-relevant attribute an entity (e.g., a network) possesses. In this work, we focus on improving three previously proposed attack graph-based security metrics: the Shortest Path metric, the Number of Paths metric, and the Mean of Path Lengths metric. Each metric attempts to answer a critical/specific question about a networks security. The Shortest Path metric attempts to answer, what is the least amount of effort an attacker can expend to violate a security policy? The Number of Paths metric attempts to answer, how many ways can an attacker violate a security policy? The Mean of Path Lengths metric attempts to answer, what is the typical effort required for an attacker to violate a security policy? Despite the questions these metrics attempt to answer, each metric has associated shortcomings. The Shortest Path metric ignores the number of ways an attacker may violate a security policy. The Mean of Path Lengths metric fails to adequately account for the number of ways an attacker may violate a security policy. The Number of Paths metric ignores the effort associated with violating a security policy. While some of the shortcomings of the Shortest Path metric and the Number of Paths metrics have been previously noted, specification of how these metrics can be used

• N. Idika is with MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 02420-9108. E-mail: nwokedi.idika@ll.mit.edu.
• B. Bhargava is with the Department of Computer Science, Purdue University, 305 N. University Street, West Lafayette, IN 47907-2107. E-mail: bb@cs.purdue.edu.

together, in concert with other metrics, to overcome their shortcomings has been overlooked in the literature.

From observing the shortcomings of previously proposed attack graph-based security metrics, we posit that network security should be measured with multiple security metrics. From this claim, at least two problems arise. The first problem is how does one combine the use of multiple security metrics? The second problem is how does one resolve conflicts that may result from using multiple security metrics? A conflict is when at least two security metrics arrive at different conclusions about what network is most secure when comparing two networks. In this work, we provide answers, in the form of an algorithm, to these questions for attack graph-based security metrics.

The contributions of this paper are as follows:

- detail the shortcoming of the Mean of Path Lengths metric,
- propose a suite of attack graph-based security metrics, and
- specify an algorithm for combining the usage of attack graph-based security metrics when evaluating the security of two networks to determine which is most secure.

In the next section, we explain the attack graph and specify the type of attack graph used in this paper. Section 3 describes related work, which includes analysis of the Shortest Path metric, the Number of Paths metric, and the Mean of Path Lengths metric. In Section 4, we propose a new complimentary set of attack graph-based security metrics. Section 5 describes how these attack graph-based metrics can be effectively used together. In Section 6, we apply our approach to two instances of attack graphs. Section 7 gives results from our simulation. Finally, in Section 8, we conclude and give directions for future work.

2 THE ATTACK GRAPH

The attack graph is a concise representation of all the ways an attacker may compromise a security policy through leveraging dependencies among known vulnerabilities. The attack graph is derived from a network model description that consists of at least the following elements: extant vulnerabilities on hosts, host connectivity, and usually at least one security policy. Extant vulnerabilities may be discovered through searching online vulnerability repositories (e.g., [2]) and/or using vulnerability scanners. Note that a vulnerability scanner does not have to identify vulnerabilities that correspond to vulnerabilities found in an online repository. Network connectivity may be determined by using firewall rules and a tool like netstat [3]. The security policies may be obtained from the Chief Security Officer (CSO) of the organization for which the attack graph is being used. There is an assortment of representations for the attack graph [4]. We use the hybrid-dependency representation of the attack graph [4]. In the hybrid-dependency graph, vulnerabilities and conditions appear as nodes in the attack graph. Vulnerabilities are described by their postconditions and preconditions. Edges in the hybrid-dependency graph represent the relationship between conditions and vulnerabilities. An edge that goes from a condition node to a vulnerability node shows that
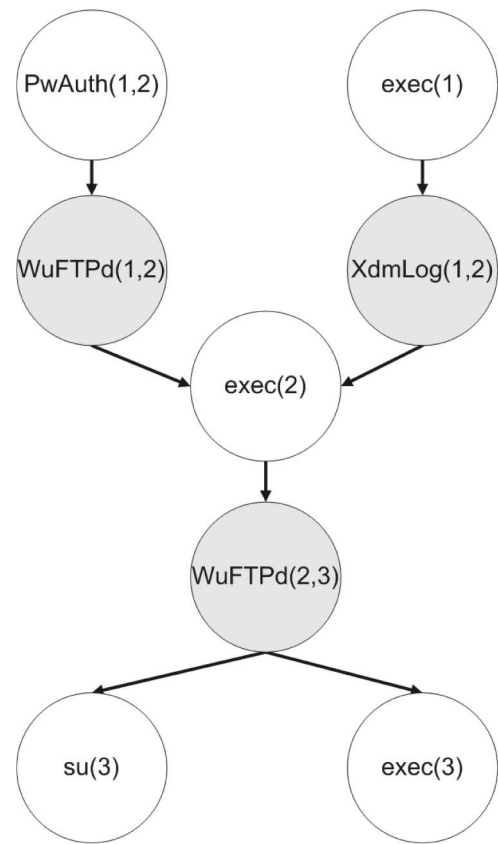

Fig. 1. An Example Attack Graph.

the condition is a precondition of the vulnerability. An edge going from a vulnerability node to a condition node shows that the condition node is a postcondition of the vulnerability. Multiple postcondition nodes for a vulnerability are to be interpreted as a disjunction of postconditions. Multiple precondition nodes for a vulnerability are to be interpreted as a conjunction of preconditions.

Fig. 1 is a simplified example from [4]. There are three hosts in this network numbered from one to three. The attack graph in Fig. 1 corresponds to a network with a security policy that states that a user on host 1 should not be able to obtain exec (i.e., execute) or su (i.e., superuser) privileges on host 3. PwAuth represents the ability to authenticate via the PwAuth program. XdmLog represents the X window display manager (xdm) login attack. WuFTPd represents an attack on the FTP server software wuarchive-ftpd. In Fig. 1, the attacker could start from two different initial states: PwAuth(1,2) or exec(1). From PwAauth(1,2), the attacker can then leverage the WuFTPd(1,2) vulnerability to obtain exec privileges on host 2 (i.e., exec(2)). Alternatively, from the initial state of exec(1), the attacker could use the XdmLog(1,2) vulnerability to reach exec(2). From exec(2), the attacker can use the WuFTPd(2,3) vulnerability to reach either goal state (i.e., su(3) or exec(3)).

As a vulnerability assessment tool, the attack graph can help an organization determine its security posture. For instance, an organization that uses the flaw hypothesis methodology [5], may use an attack graph to generate its hypotheses of how an attacker may penetrate its network. Based on these discovered paths, an organization can develop

approaches to mitigate risk. Risk mitigation takes the form of implementing countermeasures. If a security engineer uses attack graph-based security metrics, the security engineer can employ a strategy that will guide countermeasure selection [6] or compare the security of two network configurations under consideration. However, to perform these tasks effectively, a security engineer must know the instances when these metrics may produce erroneous results. We illustrate these instances in the Section 3.2.

3 RELATED WORK

In this section, we explain the intuition and semantics underlying five previously proposed attack graph-based security metrics. We categorize the security metrics broadly into nonpath analysis security metrics and path analysis security metrics.

3.1 Nonpath Analysis Attack Graph-Based Security Metrics

Nonpath analysis attack graph-based security metrics do not take into account the attributes of the attack paths an attacker must traverse to violate a security policy. Such attack graph-based security metrics are not the focus of this work. We include them for completeness.

3.1.1 The Network Compromise Percentage Metric (NCP)

The NCP metric is a security metric that Lippmann et al. proposed in [7]. This metric indicates the percentage of network assets an attacker can compromise. While the definition of compromise can be flexible to suit one's situation, Lippmann et al. defined a host compromise as the attacker attaining user-level or administrator-level access on a host. The more machines that are compromised, the higher the NCP value. Hence, the security engineer's goal is to minimize the NCP metric. The NCP metric is given in (1)

$$NCP(G) = 100 \times \frac{\sum_{c \in C \subseteq H} c.v}{\sum_{h \in H} h.v}. \qquad (1)$$

Let C be the subset of total hosts H that the attacker is able to compromise. The data member v represents the asset value associated with a host. NCP integrates coarse changes in network security. That is, if there is an increase or decrease in the number of asset-having hosts that are deemed compromised, the NCP metric will reflect this change in security. The NCP metric was proposed for attack graphs that are not goal oriented. In such attack graph analyses, the attacker has no specific target. The attacker's objective in this instance is to obtain as much network assets as possible. The attack graphs, we consider in this work assumes that the attacker is attempting to reach a specific machine or set of machines. Because all the asset value would be concentrated at some location in the network, the NCP metric would provide little insight into how the network security improves or degrades with the addition or removal of vulnerabilities in the network.

3.1.2 The Weakest Adversary Metric

Pamula et al. propose the Weakest Adversary metric in [8]. The Weakest Adversary metric is similar to the Shortest Path metric in that it attempts to express the security of the network in terms of the weakest part of the network. The intuition of the metric is that one's network is no stronger than the weakest adversary, that is, the adversary with the weakest set of capabilities. Weakness of an adversary is correlated with the initial attributes of an attack graph. Each attack graph has some set of initial attributes that allows for the realization of a security policy violation. If comparing the security of two networks, the network requiring a weaker set of initial attributes to compromise the network is deemed less secure. A set of initial attributes is deemed weaker than another set of initial attributes if it is a proper subset of the other set of initial attributes. Alternative relations could be defined for determining which of two networks has a weaker set of initial conditions. The Weakest Adversary metric is given in (2)

$$WA(G) = \{W | W \subseteq A \land \gamma(W) \leq \gamma(W')\}. \qquad (2)$$

A represents the set of initial conditions that give rise to the successful attack paths. γ is the function the security engineer would have to define in order to have a $\leq$ relation between subsets of A. In this work, we assume that attacker initial conditions are known and set by the security engineer.

3.2 Path Analysis Attack Graph-Based Security Metrics

Attack graph-based security metrics that describe the attack paths in the attack graph are considered path analysis security metrics. Because attackers reach their objective by following attack paths, we assert that understanding these attack paths is critical to network security. We take a practical quantitative approach to assessing these security metrics. This approach differs from attack graph-based security metrics requiring the usage of probabilities of successful attack [9], [10], [11]. Such approaches rely on probabilistic parameters that are currently infeasible to attain in practice. Another approach used in attack graph-based security metrics is to assign complexity values to vulnerabilities in the network to account for differences in difficulty in exploiting various vulnerabilities [12]. A standard such as the Common Vulnerability Scoring System (CVSS) [13] may be used to provide guidance in scoring vulnerabilities. Regardless, all such methodologies produce complexity values that are qualitative. Because such values exist on an ordinal scale, arithmetic and algebraic operators cannot be used to manipulate these values to arrive at a mathematically sound result.

In order to maintain the usage of arithmetic and algebraic operators, we use a quantitative definition of attack path complexity. In this paper, we take complexity to be the number of vulnerabilities an attacker must exploit to violate a security policy. If an attacker exploits three vulnerabilities to violate a security policy, irrespective of the difficulty of each vulnerability, the attacker must at least exploit three vulnerabilities. Thus, within this context, complexity is a lower bound on the true difficulty an attacker would experience in violating a security policy. If there is no ground truth regarding the complexity of vulnerabilities and arithmetic/algebraic equations are used to arrive at a result, our definition of complexity ensures that a mathematically
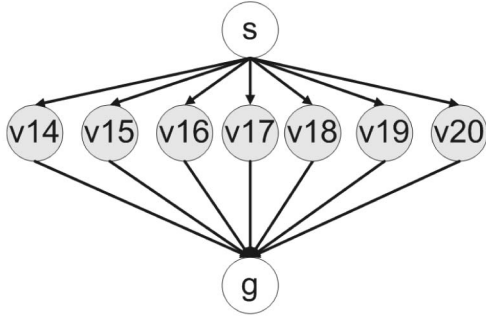
Fig. 2. Attack Graph G_i with vulnerabilities 14 through 20.

sound result is reached. In the remainder of this section, we assess path analysis security metrics that we will later combine with our proposed suite of security metrics.

3.2.1 Shortest Path Metric

The Shortest Path metric represents the length of the smallest attack path [14], [15]. The smallest attack path has the shortest distance from an attacker's initial state to the attackers desired goal state (i.e., where the security violation occurs). The length function that determines the distance is dependent on the security engineer performing the attack graph analysis. The length of an attack path may be the number of conditions, the number of exploits, or the number of conditions and exploits that start from the attacker's initial state and proceeds in series to the attacker's goal state. In this work, length is defined to be the number of exploits an attacker encounters en route to the goal state. The intuition underlying the Shortest Path metric is that from the perspective of the attacker, given the option of different steps the attacker can take to violate a security policy, the attacker will choose the series of steps that require the least amount of effort. In other words, the Shortest Path metric assumes the attacker is interested only in using the least amount of effort to reach the goal state. Effort exerted by an attacker has been represented by assigning an estimated amount of required time or resources to exploit vulnerabilities [16], [17]. Resources of an attacker may include, but are not limited to, tenacity, skills, and money [10]. These resources affect an attackers ability to penetrate a network. However, a sound mechanism for deriving attacker effort, in terms of time or resources, remains an open problem. The formalization of the Shortest Path metric is presented in (3)

$$SP(G) = \min(l(p_1), l(p_2), \ldots, l(p_k)). \qquad (3)$$

Each p_i is an attack path from the attack graph G. The function l gives the length of the attack path p_i. The function l is what the security engineer defines in the analysis process. In this work, l is the number of exploits found on an attack path. When comparing two networks, the network with the shortest attack path is the network that is less secure.

In [15], Ortalo et al. denote shortcomings of the Shortest Path metric. A noted major shortcoming is that this metric gives no indication of the number of shortest paths that may exist in a network. This shortcoming's implication is that by using the Shortest Path metric, a security engineer may arrive at an erroneous result. Assume there are two potential
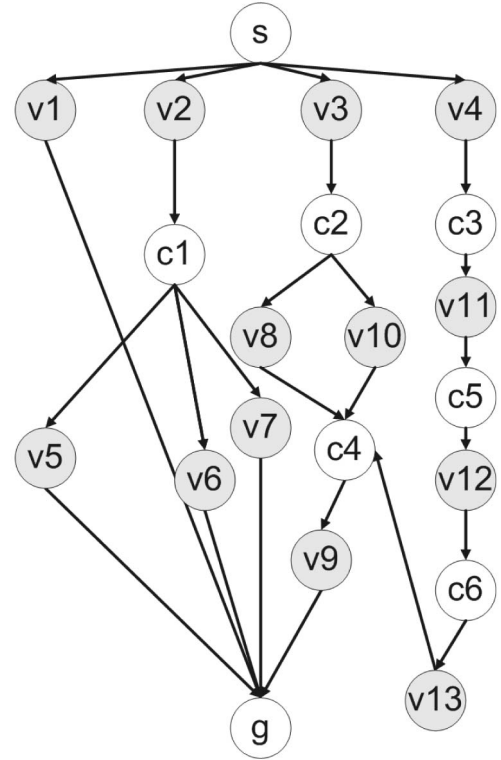

Fig. 3. Attack Graph G_j with vulnerabilities 1 through 13.

network configurations S_i and S_j that a security engineer is considering deploying, and the security engineer wants to assess the security of these two systems to determine which network to deploy. Let the attack graphs G_i and G_j in Figs. 2 and 3 correspond to the attack graphs generated for S_i and S_j, respectively. Thus, when we state G_i is less secure than G_j, then S_i is less secure than S_j. These examples were carefully chosen to obviate the differences in security between the two underlying networks. This choice is essential because it illuminates the expected outcome of comparing the two attack graphs: G_j is more secure than G_i. If the conditions in attack graphs are taken to represent hosts in a network, given our definition of complexity, G_i is intuitively less secure than G_j. When conditions correspond to hosts, G_i corresponds to a network where the attacker has seven different ways of directly violating a security policy in a single attack step. The network corresponding to G_j has a single path where the attacker may directly violate a security policy. Every other attack path in G_j requires the attacker to compromise at least one machine in the network prior to violating a security policy. The attacker's goal is to obtain condition g by exploiting the given vulnerabilities from condition s to condition g. Any of the paths of G_i produces a shortest path value of one exploit. G_j shows that the path s, v_1, g produces its shortest path value. Thus, $SP(G_i) = SP(G_j) = 1$ exploit. However, structurally, these attack graphs are relevantly distinct.

With the exception of its shortest path, G_j has paths that are strictly greater than those in G_i. Recursive application of the Shortest Path metric on subsets of attack paths of the attack graphs being compared suggests that S_i is less secure and not equivalent to S_j. Hence, the traditional application

of the Shortest Path metric could lead to an erroneous result. Erroneous results of this form highlight the coarseness of the Shortest Path metric.

The Shortest Path metric is effective for determining coarse degradation of a network's security. This effect could be the result of a new vulnerability that allows an attacker to violate a network's security in fewer steps. Because the Shortest Path metric lacks sensitivity, a security engineer could have difficulty determining the effect countermeasures can have on a network's security. For instance, if G_j gradually transformed into G_i, due to improper countermeasure selection, the Shortest Path metric would indicate that over this entire period the security of the system represented by these attack graphs is unchanged, even though the security of G_j is degrading to the security level of G_i. Similarly, the Shortest Path metric does not detect subtle improvements in security either. If G_i gradually transforms into G_j the Shortest Path metric would suggest the security of the represented system is the same over the entire transformation. This drawback suggests that this metric is not sensitive enough to be used for real-time network security evaluation independently.

A corollary of the Shortest Path metric is that longer attack paths are more secure than shorter attack paths. While this corollary suggests that paths requiring more effort is more secure, it also suggests that, under certain circumstances, having more vulnerabilities in a network could be more secure than having less vulnerabilities. Such a counterintuitive perspective is useful when completely removing certain vulnerabilities is an inviable option. In this situation, a security engineer's goal changes from removing vulnerabilities to increasing the number of vulnerabilities in the network in order to increase the effort an attacker must expend to reach his target. For instance, if a security engineer has to decide between two Web servers WS_1 and WS_2 to deploy, the security engineer may choose the server that has more vulnerabilities. For instance, assume that WS_1 allows full access to the administrator panel if the attacker, through forceful browsing, can identify the administrator directory. WS_2 may be susceptible to forceful browsing as well; however, it may require authentication to access the administrator panel. Now, further assume that the authentication routine also has a known vulnerability that allows an attacker to provide special credentials that will provide the attacker with full access to the administrator panel. If the security engineer must pick from WS_1 and WS_2 , the security engineer should choose WS_2 according to the Shortest Path metric. This choice stems from an attacker needing to execute two actions to be successful: having to discover the administrator panel through forceful browsing and having to find out the special credentials needed to access the administrator panel. With WS_1, the attacker could obtain this access using only forceful browsing.

3.2.2 Number of Paths Metric

The Number of Paths metric is a value that represents the number of ways an attacker can leverage existing dependencies among vulnerabilities to violate a network's security policy [15]. The Number of Paths metric is one that is designed to express how exposed a network is to

attack. This security metric expresses the number of attack paths that exist within a given attack graph. The intuition is that if the attacker has more ways to achieve the goal of violating a network's security policy, the attacker has a better chance of accomplishing this objective without being detected. The equation for this metric is given in (4)

$$NP(G) = |p_1, p_2, \ldots, p_k|. \quad (4)$$

Each p_i is an attack path of G. If P is the set of all attack paths in G, then the Number of Paths metric is the cardinality of this set. If we compare two attack graphs of two network systems, the attack graph with the larger number of paths is considered less secure. More attack paths translate to more opportunities for an attacker to violate a network's security policy.

A drawback of this approach is that the attack effort is not included in this metric. While one network may have fewer attack paths than another network, it may not be more secure. For instance, if an attack graph G_x has 20 attack paths and another attack graph G_y has one attack path, G_y may not necessarily be more secure than G_x. While G_x has 20 attack paths, each attack path could require effort that is 25 times greater than the effort required for the single attack path in G_y. However, there is no known way for making such quantitative assertions regarding effort in practice.

In [15], Ortalo et al. note that the Number of Paths metric is overly sensitive and unreliable. Although, the sensitivity of the Number of Paths metric is negatively regarded in [15], we maintain that this metric's sensitivity makes it useful for real-time network evaluation. This metric's sensitivity has the ability to detect fine granular changes in network security that the Shortest Path metric fails to detect. When the number of paths greatly outnumbers the number of hosts in a network, to enhance human comprehension, a security engineer could benefit from reducing the attack graph complexity before performing analysis on the attack graph [18], [19], [20], [4]. However, under such reductions care should be taken when performing hardening. A common attack graph reduction method is to treat all hosts in the same protection domain with the same reachability as a single node. If all hosts in the same protection domain are appropriately hardened, then the number of paths would decrease as a result of implementing this countermeasure. However, if countermeasures are applied to a proper subset of the hosts in the protection domain, then the effect of the countermeasures will leave the attack graph unchanged.

3.2.3 Mean of Path Lengths Metric

In [19], Li and Vaughn mention the Average Path Length metric. No detailed analysis of the metric is given. Moreover, no guidance is provided for how to use this metric. We therefore, provide our own interpretation and refer to the metric as the Mean of Path Lengths metric.

The Mean of Path Lengths metric represents the typical path length by obtaining the arithmetic mean for all path lengths. It gives an expected effort an attacker may expend to violate a network security policy. This metric is relevant because an attacker may not have the same view of the known vulnerabilities as the security engineer. For instance, this lack of knowledge could cause the attacker to choose a

path that is not the shortest path. Alternatively, an attacker may take a path different from the shortest path because the attacker could assume that the security engineer is using a shortest path analysis. With this knowledge, the attacker would avoid the shortest path because the path is likely to receive attention in the form of network activity monitors. Another reason an attacker may take a path that is not the shortest path is because the attacker's skill set may be better suited for a path that requires more effort. For instance, if the attacker is an experienced Windows hacker, but an inexperienced Linux hacker, and the network has machines of both types, the attacker may choose a seemingly circuitous route to avoid Linux machines in order to violate a security policy.

The Mean of Path Lengths metric has the ability to capture changes that occur in the network that either increase or decrease security levels. Because this mean value is computed over the entire attack graph, any degradation that results in shorter path lengths will effect the mean path length if no other path increases in length to offset the degradation. Given its attributes, the Mean of Path Lengths metric is useful for monitoring network security as well network hardening. The equation is given below

$$MPL(G) = \frac{\Sigma_i l(p_i)}{NP(G)}.$$ (5)

Despite the potential benefits this metric provides, the Mean of Path Lengths metric has some shortcomings. If vulnerabilities 15 through 20 are removed from the attack graph in Fig. 2, the resulting attack graph would produce the same mean of path lengths as the original attack graph. In short, the improvements to the network are not being captured by the Mean of Path Lengths metric. Another shortcoming with this metric is that the mean attack path length for a network may increase as the number of vulnerabilities increase in the network. This phenomenon arises because the attacker, via increased vulnerabilities, is provided with more circuitous routes to reaching the target.

4 EXTENDING ATTACK GRAPH-BASED SECURITY METRICS

While the above mentioned security metrics can be useful if used appropriately, if any metric is used in isolation, one may arrive at a misleading conclusion. The Shortest Path metric can be too coarse. The Number of Paths metric does not capture attacker effort. The Mean of Path Lengths metric does not detect changes that do not effect the mean path length. If these metrics are used together, they can give a more comprehensive measure of security. We detail how in Section 5. However, in this section, we propose a complimentary set of metrics to assist a security engineer in determining more relevant properties of the network to determine its security. The metrics we propose are the following: the Normalized Mean of Path Lengths metric, the Standard Deviation of Path Lengths metric, the Mode of Path Lengths metric, and the Median of Path Lengths metric.

4.1 Normalized Mean of Path Lengths Metric

The Normalized Mean of Path Lengths metric is the Mean of Path Lengths metric divided by the Number of Paths

metric. The identified shortcomings of the Mean of Path Lengths metric stems from its failure to appropriately take into account the Number of Paths metric. The Normalized Mean of Path Lengths metric addresses this issue by normalizing the Mean of Path Lengths by the number of paths in the attack graph. Through this approach, we can detect fine granular improvements and degradations in network security. Moreover, this metric provides an approach for interpreting two attack graphs that have a different number of attack paths. For instance, if vulnerabilities 15 through 20 are removed from attack graph G_i, the new attack graph would be deemed more secure than the original attack graph by the Normalized Mean of Path Lengths metric. In comparing two attack graphs, the attack graph with the smaller Normalized Mean of Path Lengths metric is deemed less secure. The equation for this metric is given below

$$NMPL(G) = \frac{MPL(G)}{NP(G)}.$$ (6)

4.2 Standard Deviation of Path Lengths Metric

The Standard Deviation of Path Lengths metric, when added and subtracted from the Mean of Path Lengths metric, gives a range containing typical attack path lengths. These typical attack path lengths have path lengths that are within one standard deviation of the mean path length. The Standard Deviation of Path Lengths metric may also reveal attack paths of interest. If, for instance, a path length is two standard deviations below the Mean of Path Lengths metric, this path may deserve the attention of the security engineer. If, however, a path length is two standard deviations above the mean path length, this finding may suggest that this path may not require the attention other attack paths may require. The equation for this metric is given below

$$SDPL(G) = \sqrt{\frac{\Sigma_i (l(p_i) - MPL(G))^2}{NP(G)}}.$$ (7)

4.3 Mode of Path Lengths Metric

The Mode of Path Lengths metric gives the attack path length that occurs most frequently. This metric represents another meaning of typical. In this context, typical refers to most frequent. If the security engineer is unable to determine the likelihood of an attacker traversing any attack path, the security engineer may rely on the principle of insufficient reason [21] to assign an equal probability to each attack path. The Mode of Path Lengths metric suggests a likely amount of effort an attacker may encounter. The Mode of Path Lengths metric is not as dynamic as the Mean of Path Length metric in response to security events. However, unlike the Mean of Path Lengths metric, the Mode of Path Lengths metric may not be as prone to being effected by outlier values. In the equation below, f is a function that identifies the $l(p_i)$ that occurs most frequently of the $k = NP(G)$ values

$$MoPL(G) = f(l(p_1), l(p_2), \ldots, l(p_k)).$$ (8)

4.4 Median of Path Lengths Metric

The Median of Path Lengths metric identifies the path length that is at the middle of all the path length values.

This value is useful because the path lengths can be skewed and, therefore, the Mean of Path Lengths metric may not appropriately indicate the typical path lengths in the attack graph. A very large path length, and similarly a very small path length, in an attack graph can bias the Mean of Path Lengths metric. The median path length helps the security engineer determine how close the mean attack path length is to the middle of all attack path lengths. The Median of Path Lengths metric may also provide a guide for where to focus network hardening efforts for the security engineer. For instance, the security engineer may choose to pay close attention to attack paths with path lengths equal to or below the median path length. In the equation below. Note that $l(p_i)_q \le l(p_j)_{q+1}$, states that the path length of path p_i at position q is shorter than p_j and precedes p_j, which is at position $q + 1$. k is the number of attack paths in the attack graph G.

$$MePL(G) = \begin{cases} l(p_i)_{\lceil \frac{k}{2} \rceil} : & \text{k is odd} \\ \frac{1}{2}\left(l(p_i)_{\frac{k}{2}} + l(p_j)_{\frac{k}{2}+1}\right) : & \text{k is even} \end{cases} \quad (9)$$

4.5 Addressing Scalability

The complexity of each metric is $O(NP(G)longestPath(G))$. That is, each metric's complexity is in big-oh of the product of the number of paths in attack graph G and the longest attack path in G. Because $NP(G)$ can grow exponentially with respect to the number of hosts in the network, some approaches for reducing how much of G needs to be traversed to obtain a metric's value may be important.

If metrics are being used together as it is suggested in this work (see Section 5), then the attack graph can be traversed once and numerical values can be obtained for each attack path that would represent the attack path length. Thus, using an array of path lengths, for example, all path analysis metrics can be used on this 1D array instead of an attack graph (2D array). The size of the array would correspond to the number of paths in the attack graph, and each element of the array would correspond to the attack path length. After traversing the initial attack graph to obtain the array, computing the metric values will be in $O(NP(G))$. Developing methods for more efficient methods for metric computation is left for future work.

5 USING MULTIPLE SECURITY METRICS

In this section, we propose a methodology for using these metrics together harmoniously. We assert that no one security metric will divulge all there is to know about a network's security. Given this assertion, an important goal in evaluating the security of two networks is to have a method for combining the usage of appropriate metrics to reach a decision about which network is most secure. We explain how to achieve this goal in this section.

5.1 Decision Metrics

Decision metrics are the security metrics that when comparing two attack graphs of two networks, makes a determination about which network is more secure. Decision metrics are security metrics that should be applied first. The decision metrics discussed in this paper are: the Shortest Path metric, the Number of Paths metric, the

TABLE 1
Classification of Attack Graph-Based Security Metrics

Type	Metric
Decision	SP Shortest Path
	NP Number of Paths
	NMPL Normalized Mean of Path Lengths
	NCP Network Compromise Percentage
	WA Weakest Adversary
Assistive	MPL Mean of Path Lengths
	SDPL Standard Deviation of Path Lengths
	MoPL Mode of Path Lengths
	MePL Median of Path Lengths

Normalized Mean of Path Lengths metric, the Network Compromise Percentage metric, and the Weakest Adversary metric. These metrics are also shown in the top row of Table 1. Our algorithm uses path analysis metrics.

When evaluating two attack graphs, G_1 and G_2 to determine which is most secure, G_1 is strictly less secure than G_2 if G_1 has the shortest attack path length, the most number of attack paths, and the smaller normalized mean of path lengths. In other words, G_2 strictly dominates G_1. In the cases where G_2 cannot be determined to be strictly more secure than G_1 and vice versa via decision metrics, an approach for reaching a decision may be creating a total ordering of priority on questions a security engineer would like to answer: 1) Which attack graph produces the shortest attack path requiring the most effort? 2) Which attack graph gives the attacker the least number of ways of violating a security policy? 3) Which attack graph produces attack paths that typically require more effort to violate a security policy given the number of ways of doing so? The first, second, and third questions correspond to the Shortest Path metric, the Number of Paths metric, and the Normalized Mean of Path Lengths metric, respectively. Once this priority structure is established, the attack graph satisfying the question deemed most important is considered more secure. Alternatively, the security engineer could assume each question is equally important. In this scenario, the attack graph satisfying the most questions above is deemed more secure. The security engineer could also vary the weights of importance associated with these questions.

This approach to network security evaluation is specified in *compareGraphs* in Algorithm 1. The security engineer would supply Algorithm 1 with the two attack graphs to compare, the set of metrics deemed most important to evaluate the attack graphs, and a minimum frequency value t (detailed later in Section 5.2) in case assistive metrics are required. The algorithm starts by examining each metric of interest, and applying the metric to the attack graphs being compared (lines 1-9). The *applyMetric* function (lines 3, 5, 7) uses a *decide* function to determine which attack graph is most secure (line 1, Algorithm 2). In the *decide* function, Algorithm 3, a sanity check is done initially (lines 1-3). The *decide* function then determines which attack graph is most secure with respect to metric m (lines 4-9).

Algorithm 1. *compareGraphs* function: Algorithm for Using Multiple Metrics to Evaluate Two Attack Graphs
Require: G_1, G_2 {attack graphs to be compared}
Require: M {the set of security metrics to be used for attack graphs}

Require: t {threshold value for Mode of Path Lengths metric}
Require: R_d {the set of results from applying decision metrics to both attack graphs}
  1: **for all** $m \in M$ **do**
  2:    **if** m equals SP **then**
  3:      $R_d \leftarrow applyMetric(G_1, G_2, R_d, m, null, >)$
  4:    **else if** m equals NP **then**
  5:      $R_d \leftarrow applyMetric(G_1, G_2, R_d, m, null, <)$
  6:    **else if** m equals $NMPL$ **then**
  7:      $R_d \leftarrow applyMetric(G_1, G_2, R_d, m, null >)$
  8:    **end if**
  9: **end for**
10: **if** $isStrictlyDominated(R_d, G_1, G_2)$ **then**
11:    **return** $(R_d,$ {"strictly dominated"})
12: **else if** $isMajorityDominated(R_d, G_1, G_2)$ **then**
13:    **return** $(R_d,$ {"majority dominated"})
14: **else if** $isEqual(R_d)$ **then**
15:    **return** $(R_d,$ {"all are equal"})
16: **end if**
17: **return** $enlistAssistiveMetrics(G_1, G_2, R_d, t)$

Algorithm 2. $applyMetric$ function
Require: G_1, G_2 {attack graphs to be compared}
Require: R {result set}
Require: m_1 {metric to apply to the two attack graphs}
Require: m_2 {assistive metric}
Require: c {relational operator}
  1: $r_0 \leftarrow decide(G_1, G_2, m_1, c)$
  2: $r_1 \leftarrow m_1$
  3: **if** m_2 does not equal $null$ **then**
  4:    $r_2 \leftarrow m_2$
  5: **end if**
  6: **return** $R.add(r)$ {r is compromised of r_0, r_1 and r_2}

Algorithm 3. $decide$ function
Require: G_1, G_2 {attack graphs to be compared}
Require: m {security metric to apply to G_1 and G_2}
Require: c {relational operator}
  1: **if** G_1 equals $\emptyset$ or G_2 equals $\emptyset$ **then**
  2:    **return** "$incomparable$"
  3: **end if**
  4: **if** $m(G_1) \, c \, m(G_2)$ **then**
  5:    **return** $G_1.name$
  6: **else if** $m(G_1)$ equals $m(G_2)$ **then**
  7:    **return** ""
  8: **end if**
  9: **return** $G_2.name$

When assistive metrics are not being used in *compareGraphs*, the m_2 parameter of *applyMetric* will be null (lines 3, 5, 7, Algorithm 1). Once the for loop completes in Algorithm 1, the algorithm determines whether an attack graph strictly dominates the other attack graph on each metric (lines 10-11). It then determines whether an attack graph dominates the other attack graph on a majority of the metrics used (lines 12-13, Algorithm 1). The last check performed is to determine if the two attack graphs are equal (lines 14-15). The *isStrictlyDominated* and *isMajorityDominated* functions are specified in

Algorithms 4 and 5, respectively. If further analysis is required, then this analysis is obtained from assistive metrics (line 17, Algorithm 1).

Algorithm 4. $isStrictlyDominated$ function
Require: R {result set}
Require: G_1, G_2 {attack graphs to be compared}
  1: $v_1 \leftarrow countFrequency(G_1.name, R)$
  2: $v_2 \leftarrow countFrequency(G_2.name, R)$
  3: **if** $(v_1 equals R.size)$ **or** $(v_2 equals R.size)$ **then**
  4:    **return** true
  5: **end if**
  6: **return** false

Algorithm 5. $isMajorityDominated$ function
Require: R {result set}
Require: G_1, G_2 {attack graphs to be compared}
  1: $v_1 \leftarrow countWeightedFrequency(G_1.name, R)$
  2: $v_2 \leftarrow countWeightedFrequency(G_2.name, R)$
  3: **if** $(v_1 > v_2)$ **or** $(v_2 > v_1)$ **then**
  4:    **return** true
  5: **end if**
  6: **return** false

In the cases, where conclusions cannot be drawn from examining the Number of Paths metric and the Shortest Paths metric alone, the Normalized Mean of Path Lengths metric will usually provide a conclusive answer (see Section 7). However, in the cases when this conclusion cannot be achieved, assistive metrics are required.

5.2 Assistive Metrics

Assistive metrics serve as "drill down" metrics. These metrics help discover more security-relevant information about the nature of the attack graph. Unlike decision metrics, these metrics are not to be used independently to make determinations regarding what attack graph is most secure when comparing two attack graphs.

Assistive metrics mentioned in this paper include: the Mean of Path Lengths, the Standard Deviation of Path Lengths metric, the Mode of Path Lengths metric, and the Median of Path Lengths metric. These metrics are listed in the bottom row of Table 1. We have specified our approach of using assistive metrics in Algorithm 6. If the Shortest Path metric was used previously, then the Shortest Path metric will be applied to subsets of attack paths of the two attack graphs being compared (lines 3-9). The subset of attack paths function s creates an attack graph containing subsets of attack paths from G based on a metric m. s is specified in Algorithm 7. If s is passed an optional parameter t and the attack graph G does not have enough path lengths equal to the Mode of Path Lengths metric, the empty set is returned to indicate that G does satisfy the threshold (lines 1-3, Algorithm 7).

Algorithm 6. $enlistAssistiveMetrics$ function
Require: G_1, G_2 {attack graphs to be compared}
Require: R_d {results set from applying decision metrics on G_1 and G_2}
Require: R_a {the set of results from using assistive metrics}
Require: t {threshold value for Mode of Path Lengths metric}

```
1:  for all r.r₁ ∈ R_d do
2:    if r.r₁ equals SP then
3:      for all m ∈ {MoPL, SDPL} do
4:        if m equals MoPL then
5:          R_a ← applyMetric(s(G₁, m, t), s(G₂, m, t),
             R_a, r.r₁, m, >)
6:        else
7:          R_a ← applyMetric(s(G₁, m), s(G₂, m), R_a,
             r.r₁, m, >)
8:        end if
9:      end for
10:   else if r.r₁ equals NP then
11:     MePL'.value ← min(MePL(G₁), MePL(G₂))
12:     for all m ∈ {MePL', SDPL} do
13:       R_a ← applyMetric(s(G₁, m), s(G₂, m),
            R_a, r.r₁, m, >)
14:     end for
15:   else if r.r₁ equals NMPL then
16:     for all m ∈ {MePL', MoPL, SDPL} do
17:       MePL'.value ← min(MePL(G₁), MePL(G₂))
18:       if m equals MoPL then
19:         R_a ← applyMetric(s(G₁, m, t), s(G₂, m, t),
             R_a, r.r₁, m, >)
20:       else
21:         R_a ← applyMetric(s(G₁, m), s(G₂, m),
             R_a, r.r₁, m, >)
22:       end if
23:     end for
24:   end if
25: end for
26: if isStrictlyDominated(R_a, G₁, G₂) then
27:   return (R_d, R_a, {"strictly dominated"})
28: else if isMajorityDominated(R_a, G₁, G₂) then
29:   return (R_d, R_a, {"majority dominated"})
30: else if isEqual(R_a, G₁, G₂) then
31:   return (R_d, R_a, {"all are equal"})
32: end if
33: return (R_d, R_a)
```

Algorithm 7. s function
Require: G {attack graph to create a subgraph from}
Require: m {metric that will be used}
Require: t {optional parameter that is used for the Mode
  of Path Lengths metric}

```
1:  if t is passed in as a parameter then
2:    if isBelowThreshold(t, m(G), G) then
3:      return ∅
4:    end if
5:  end if
6:  if m equals MoPL then
7:    return G' ← keepPathsEqualTo(m(G), G)
8:  else if m equals SDPL then
9:    return G' ← keepPathsInRange(χ(G) − m(G),
       χ(G) + m(G), G)
10: else if m equals MePL' then
11:   return G' ← keepPathsBelowOrEqualTo(m(G), G)
12: end if
```

In the s function, the *keepPathsEqualTo* function returns an attack graph with paths that have path lengths equal to the Mode of Path Lengths metric (lines 6-7). The *keepPathsInRange* function returns an attack graph with paths that have path lengths that are within the range specified by the Mean of Path Lengths metric and the Standard Deviation of Path Lengths metric (lines 8-9).

The *keepPathsBelowOrEqualTo* returns an attack graph with paths that have path lengths equal to or below the modified Median of Path Lengths metric (lines 10-11). We explain the modification to the Median of Path Lengths metric later in this section.

In the *enlistAssistiveMetrics* function in Algorithm 6, when the Shortest Path metric is used to compare $s(G_1, MoPL)$ and $s(G_2, MoPL)$, the result of this expression captures the comparison of one notion of the most common amount of attack effort. The attack graph having a typical amount of attack effort that is greater with respect to the Shortest Path metric is considered most secure. When the Shortest Path metric is used to compare $s(G_1, SDPL)$ and $s(G_2, SDPL)$, the result of the expression captures the identification of the attack graph having the path of least resistance requiring the most effort among another notion of the typical attack effort.

If the Number of Paths metric was initially used to determine which attack graph was most secure (line 10), the metric will be applied to the subsets of attack paths yielded from applying the Standard Deviation of Path Lengths metric, and a modified Median of Path Lengths metric to the attack graphs using s (lines 11-14). When the Number of Paths metric is applied to the $s(G_1, SDPL)$ and $s(G2, SDPL)$, the resulting expression captures the number of typical paths in the attack graphs for a notion of typical. $MePL'$, the modified Median of Path Lengths metric, is the minimum median of the two attack graphs being compared. Because low resistance paths are undesirable from the security engineer's perspective, these are the paths most worthy of attention. When the Number of Paths metric is applied to $s(G_1, MePL')$ and $s(G_2, MePL')$, the expression captures which attack graph has the least number of paths of low resistance to attack.

If the Normalized Mean of Path Lengths metric was initially used to determine which attack graph was most secure, the metric will be applied to the subsets of attack paths yielded from applying the Mode of Path Lengths metric, the Standard Deviation of Path Lengths metric, and the modified Median of Path Lengths metric (lines 15-23). The application of the Normalized Mean of Path Lengths metric on $s(G_1, MoPL)$ and $s(G_2, MoPL)$ takes into the account the effort and the number of paths that is associated with the most frequently occurring attack path lengths. When the Normalized Mean of Path Lengths metric is used to compare $s(G_1, SDPL)$ and $s(G_2, SDPL)$, the expression captures the attack effort associated with one notion of common attack paths in the attack graphs. When the Normalized Mean of Path Lengths metric is used to compare $s(G_1, MePL')$ and $s(G_2, MePL')$, the expression captures the attack effort associated with the weakest set of attack paths.

TABLE 2
Security Metric Evaluation of G_i and G_j

Metric	G_i	G_j
SP Shortest Path	1	1
NP Number of Paths	7	7
MPL Mean of Path Lengths	1	2.5
NMPL Normalized Mean of Path Lengths	0.14	0.35
SDPL Standard Deviation of Path Lengths	0	1.1
MoPL Mode of Path Lengths	1	2
MePL Median of Path Lengths	1	2

To come to a decision about which attack graph is most secure, one can establish a total ordering for the values in R_a. The attack graph that is determined to be most secure for the element in R_a deemed most important would be considered most secure. Alternatively, an equal or varied weighting scheme could be used. *isStrictlyDominated*, *isMajorityDominated*, and/or *isEqual* will be called with the attack graphs being compared and R_a to reach a determination of which attack graph is most secure. If the usage of decision and assistive metrics fails to produce any conclusions, then a security engineer would be relegated to experience and expert opinion in determining what network is most secure.

6 ASSESSING TWO ATTACK GRAPHS

Table 2 gives the result of applying each of the path analysis metrics to attack graphs G_i and G_j. It is evident from the table that looking solely at the Shortest Path metric or the Number of Paths metric, one may end up at a potentially wrong conclusion. However, by using more security metrics, we can make greater distinctions about the two underlying networks and arrive at a more reasoned conclusion. By looking at the Median of Path Lengths metric for G_j and inspecting the path lengths above and below it, the security engineer can ascertain that it has at least three different values (one, two, and five exploits). This reveals that half of the attack paths are equal to the shortest path or are a single exploit away from being considered a shortest path. This may suggest that if the network represented by G_j is chosen, the segment of the network producing the attack paths below the median may warrant special attention. The Standard Deviation of Path Lengths metric also reveals the homogeneity of the path lengths in G_i. However, for G_j, the Standard Deviation of Path Lengths metric suggests that typical path lengths are approximately within the range of one exploit and three exploits. The result of applying the *compareGraphs* is shown in Table 3.

7 SIMULATION RESULTS

In this section, we show the results of using *compareGraphs* for two sets of randomly generated attack graphs. The number of paths in the attack graphs are uniformly distributed between 1 and 2,000 attack paths. The path lengths is uniformly distributed between 1 and 50. 1,000 randomly generated attack graphs are assigned to set G_a, and another 1,000 randomly generated attack graphs are

TABLE 3
The Result of Applying Decision
and Assistive Metrics to G_i and G_j

Security Metric Value	Most Secure
SP on G_i and G_j	G_i and G_j
NP on G_i and G_j	G_i and G_j
NMPL on G_i and G_j	G_j
SP on $s(G_i, MoPL, t)$ and $s(G_j, MoPL, t)$	G_j
SP on $s(G_i, SDPL)$ and $s(G_j, SDPL)$	G_j
NP on $s(G_i, MePL')$ and $s(G_j, MePL')$	G_j
NP on $s(G_i, NMPL)$ and $s(G_j, NMPL)$	G_j
NP on $s(G_i, MoPL, t)$ and $s(G_j, MoPL, t)$	G_j
NMPL on $s(G_i, MePL')$ and $s(G_j, MePL')$	G_j
NMPL on $s(G_i, NMPL)$ and $s(G_j, NMPL)$	G_j
NMPL on $s(G_i, MoPL, t)$ and $s(G_j, MoPL, t)$	G_j

assigned to set G_b. *compareGraphs* is called for all pairs of attack graphs belonging to distinct sets. The value two is used for the parameter t for *compareGraphs*. The results are shown in Table 4.

In Table 4, DMs and AMs refer to Decision Metrics and Assistive Metrics, respectively. The "Overall" row depicts the percentage of comparisons that Decision Metrics and Assistive Metrics can decide together. The "Strictly by DMs" row depicts the percentage of comparisons where all decision metrics used chose the same attack graph to be more secure. The "Majority by DMs" row depicts the percentage of comparisons where the majority of the used decision metrics chose the same attack graph to be most secure. The "Equal by DMs" row depicts the percentage of comparisons that were determined to be equal by the used decision metrics. The meaning of "Strictly with AMs," "Majority with AMs," "Equal by AMs" rows are the same as their "by DMs" counterparts except that they are with the usage of assistive metrics.

Table 4 shows that the combination of decision metrics used to evaluate attack graphs under consideration affect whether *compareGraphs* will be able to reach a conclusion about which attack graph is most secure. Regardless of what combination of decision metrics are used, assistive metrics help reach conclusions regarding which attack graph is most secure.

8 CONCLUSION AND FUTURE WORK

We examined previously proposed attack graph-based security metrics. We examined in-depth the shortcomings of three path-analysis attack graph-based security metrics. We proposed a complimentary suite of attack graph-based security metrics to compensate for the shortcomings of

TABLE 4
The Percentage of Compared Attack Graphs
for Which *compareGraphs* Reaches a Decision

Decision	*SP, NP*	*SP, NMPL*	*NMPL, NP*	*All*
% Overall	48.4	78	99.9	99.9
% Strictly by DMs	4	4	99	4
% Majority by DMs	0	0	0	95
% Equal by DMs	.4	0	0	0
% Strictly with AMs	10	10	.1	.1
% Majority with AMs	34	64	.8	.8
% Equal with AMs	0	0	0	0

path analysis attack graph-based security metrics. We have detailed how to use the above mentioned metrics with our proposed suite of metrics to measure the security of networks under consideration. We have shown through simulated results that in many instances, our approach for metric combination is able to decide which of two attack graphs correspond to a more secure network.

Our future work includes producing more attack graph-based security metrics. Increasing the number security metrics that provide unique security-relevant information will enhance the security engineer's ability to assess a network's security and to perform network hardening. Future work also includes developing enhanced approaches for quantitatively measuring attack path complexity. Lastly, our future work includes developing efficient means for computing attack graph-based security metrics.

ACKNOWLEDGMENTS

REFERENCES

[1] SSE-CMM, http://www.sse-cmm.org/metric/metric.asp, 2010.
[2] http://www.cve.mitre.org, MITRE CVE, July 2010.
[3] G. Vigna and R. Kemmerer, "Netstat: A Network-Based Intrusion Detection System," *J. Computer Security,* vol. 7, 1999.
[4] S. Noel and S. Jajodia, "Managing Attack Graph Complexity through Visual Hierarchical Aggregation," *Proc. ACM Workshop Visualization and Data Mining for Computer Security,* pp. 109-118, 2004.
[5] C. Weissman, "System Security Analysis/Certication Methodology and Results," Technical Report SDC SP-3728, 1973.
[6] N. Idika, B. Marshall, and B. Bhargava, "Maximizing Security given a Limited Budget," *Proc. TAPIA '09: Richard Tapia Celebration of Diversity in Computing,* Apr. 2009.
[7] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham, "Validating and Restoring Defense in Depth Using Attack Graphs," *Proc. Military Communications Conf.,* Oct. 2006.
[8] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A Weakest-Adversary Security Metric for Network Configuration Security Analysis," *Proc. Second ACM Workshop Quality of Protection,* pp. 31-38, 2006.
[9] S. Jha, O. Sheyner, and J. Wing, "Two Formal Analyses of Attack Graphs," *Proc. 15th IEEE Computer Security Foundations Workshop,* June 2002.
[10] R. Dantu and P. Kolan, "Risk Management Using Behavior Based Bayesian Networks," *Intelligence and Security Informatics,* pp. 115-126, 2005.
[11] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An Attack Graph-Based Probabilistic Security Metric," *Proc. Data and Applications Security (DAS '08),* pp. 283-296, 2008.
[12] L. Wang, A. Singhal, and S. Jajodia, "Measuring Overall Security of Network Configurations Using Attack Graphs," *Data and Applications Security XXI,* vol. 4602, pp. 98-112, Aug. 2007.
[13] P. Mell, K. Scarfone, and S. Romanosky, "Common Vulnerability Scoring System," *IEEE Security and Privacy,* vol. 4, pp. 85-89, Nov./ Dec. 2006.
[14] C. Phillips and L.P. Swiler, "A Graph-Based System for Network-Vulnerability Analysis," *NSPW '98: Proc. Workshop New Security Paradigms,* pp. 71-79, 1998.
[15] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security," *IEEE Trans. Software Eng.,* vol. 25, pp. 633-650, Sept. 1999.
[16] E. Jonsson and T. Olovsson, "A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior," *IEEE Trans. Software Eng.,* Apr. 1997.
[17] G. Schudel and B. Wood, "Adversary Work Factor as a Metric for Information Assurance," *Proc. 2000 Workshop New Security Paradigms,* pp. 23-30, 2001.
[18] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical Attack Graph Generation for Network Defense," *Proc. Computer Security Applications Conf.,* pp. 121-130, Dec. 2006.
[19] W. Li and R. Vaughn, "Cluster Security Research Involving the Modeling of Network Exploitations Using Exploitation Graphs," *Proc. Sixth IEEE Int'l Symp. Cluster Computing and Grid Workshops,* May 2006.
[20] S. Noel, M. Jacobs, P. Kalapa, and S. Jajodia, "Multiple Coordinated Views for Network Attack Graphs," *Proc. IEEE Workshop Visualization for Computer Security,* pp. 99-106, 2005.
[21] P. Dupount, "Laplace and the Indifference Principle in the 'Essai Philosophique Des Probabilits'," *Rend. Sem. Mat. Univ. Politec. Torino,* vol. 36, pp. 125-137, 1977/78.

Nwokedi Idika received the BS degree in computer science from the University of Maryland, Baltimore County in 2005. He obtained the MS and PhD degrees in 2007 and 2010, respectively, in computer science from Purdue University, West Lafayette, Indiana. His research interests include vulnerability assessment, network security, and security metrics.

Bharat Bhargava received the BE degree from the Indian Institute of Science and the MS and PhD degrees in electrical engineering from Purdue University, West Lafayette, Indiana. He is currently a professor of computer science at Purdue University. His research involves mobile wireless networks, secure routing and dealing with malicious hosts, providing security in Service Oriented Architectures (SOA), adapting to attacks, and experimental studies. His name has been included in the Book of Great Teachers at Purdue University. Moreover, he was selected by the student chapter of ACM at Purdue University for the Best Teacher Award. He is a fellow of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.