CERIAS Tech Report 2010-23

Characterizing and Aggregating Attack Graph-based Security Metrics

by Nwokedi C. Idika
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

CHARACTERIZING AND AGGREGATING ATTACK GRAPH-BASED SECURITY METRICS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Nwokedi C. Idika

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August, 2010

Purdue University

West Lafayette, Indiana

This dissertation is dedicated to my family, my friends, and the Meyerhoff Scholarship Program.

ACKNOWLEDGMENTS

I would like to recognize individuals who helped me put this document together. I would like to recognize my advisor, Professor Bharat Bhargava for his assistance throughout the Ph.D. process. I would like to recognize Professor Aditya Mathur, my initial advisor, for encouraging me to come to Purdue University and for providing encouragement once I arrived at Purdue University. I would like to recognize my Ph.D. committee for providing useful insights that caused me to think more deeply about my work. I would like to recognize Eugene Spafford for introducing me to the topic of Security Metrics and giving me insight on how to conduct research. I would like to recognize Professor Brandeis Hill Marshall and Professor M. Brian Blake for looking at early drafts of this dissertation.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AR	Attack Resistance
EX	Exploitability
KCA	K-step Condition Accumulation
MDT	Mean Downtime
MPL	Mean of Path Lengths
MoPL	Mode of Path Lengths
MTFF	Mean-Time-to-First-Failure
MTTB	Mean-Time-To-Breach
MTTF	Mean-Time-To-Failure
NCP	Network Compromise Percentage
NMPL	Normalized Mean of Path Lengths
NP	Number of Paths
SDPL	Standard of Deviation of Path Lengths
SP	Shortest Path

GLOSSARY

attack graph	an abstraction representing the ways an attacker can use the interdependencies among vulnerabilities to violate a security policy
attack path	a sequence of steps that starts from an attacker's initial state to the attacker's goal state (security policy violation) in an attack graph
breach	a security policy violation
countermeasure	any action performed to remove at least one vulnerability from a system
entity	a computer system, a process, a person, or a collection of such entities
exploit	an instance of a vulnerability specifying the preconditions and postconditions of the vulnerability with details from the network under study
hardening	providing protection to an entity
security	the process of providing confidentiality, integrity and availability to an entity according to some policy
security control	a countermeasure or safeguard
security metric	a value produced from measuring identifiable entity attributes that affect physical, personnel, IT, or operational security
security policy	a requirement that specifies the secure states of the network
vulnerability	a weakness in the system that is described by its preconditions and postconditions

ABSTRACT

Idika, Nwokedi C. Ph.D., Purdue University, August, 2010. Characterizing and Aggregating Attack Graph-based Security Metrics. Major Professor: Bharat K. Bhargava.

An attack graph is an abstraction that represents the ways an attacker can violate a security policy by leveraging interdependencies among discovered vulnerabilities. Attack graph analyses that extract security-relevant information from the attack graph are referred to as attack graph-based security metrics. Although a number of attack graph-based security metrics have been proposed in the literature, there has been no analysis of how these security metrics behave in response to security incidents. In this dissertation, we examine how attack graph-based security metrics behave in response to increased network vulnerabilities under heterogeneous network models. From this analysis, we identify opportunities for using equations that characterize particular attack graph-based security metrics avoiding the costly processing of attack graphs.

Security is recognized to be a multidimensional entity. However, all proposed attack graph-based security metrics have been unidimensional. In this dissertation, we provide an approach for aggregating the capabilities of existing attack graph-based security metrics with our proposed suite of attack graph-based security metrics.

Lastly, we specify an algorithm for network hardening given a limited budget. Given a set of network vulnerabilities and a set of candidate countermeasures to implement, a network administrator is to choose the set of countermeasures that optimize security given a limited budget. Our algorithm produces sets of countermeasures that optimize security with respect to a set of attack graph-based security metrics while staying within budget.

1. INTRODUCTION

Security metrics produce assessments of an entity's ability to provide itself with confidentiality, integrity, and availability. These assessments are values derived from measuring security-relevant entity attributes. A security-relevant attribute is one that affects an entity's ability to provide itself confidentiality, integrity, or availability. When the entity being measured is an organization, the values produced can be referred to as enterprise security metrics.

INFOSEC affirms that enterprise security metrics is a top 8 security research priority [1]. This ranking is due, in part, to the compelling promise of enterprise security metrics. With these security metrics, an organization could (1) determine whether its security is improving over time, (2) determine its return on investment on implemented security controls, and (3) determine how well its security compares to other similar organizations [2]. Although there are no universally accepted security metrics for addressing issues (1) - (3), progress has been made toward this end [3].

In general, security metrics are used to help achieve security goals. The goals of security are to prevent, detect, and recover from *attacks* [4]. Attacks, which are actions that violate some security policy, are inextricably linked to the goals of security. Given the importance of attacks to security, a means for representing the ways an attacker can successfully attack a system is invaluable to security evaluation.

The attack graph represents the possible ways an attacker can violate a security policy. An attack graph can be generated from network configuration details and known vulnerabilities within the network. The attack graph depicts how an attacker could leverage dependencies among vulnerabilities to violate a security policy. If an attacker violates a security policy by first exploiting a Secure Shell Daemon (sshd) vulnerability on machine m_1 and then exploits a Remote Shell Daemon (rshd) vulnerability on machine m_2, the attacker would be taking advantage of the interdependency

between the sshd vulnerability on m_1 and the rshd vulnerability on m_2. The attackers sequence of actions would constitute an attack path. In one type of attack graph where nodes correspond to hosts and edges corresponds to vulnerability exploits, this attack path could be represented by three nodes and two edges. The third node would represent the machine from which the attacker initiated the attack on m_1. Thus, an attack graph is a collection of attack paths.

Because the attack graph is derived from a network, analyzing the attack graph can reveal security-relevant properties about the network. An analysis designed to extract these security-relevant properties are referred to as *attack graph-based security metrics*. Although a number of attack graph-based security metrics have been proposed, we assert that no one proposed security metric fully describes a network's security. Despite having an array of attack graph-based security metrics that measure different attributes of a network's security, there are no techniques for combining these metrics to obtain a more comprehensive view of a network's security.

1.1 Thesis Statement

Attack graph-based security metrics can be characterized and aggregated to assist network security evaluation and enhancement.

1.1.1 Characterization of Attack Graph-based Security Metrics

To characterize attack graph-based security metrics, we identify their distinguishing attributes. We are interested in three questions. Two questions are (1) how and when should attack graph-based security metrics be used, and (2) are there equations that explain the behavior of attack graph-based security metrics. These questions are derived from how attack graph-based security metrics respond to security relevant events. The security relevant events of concern in this dissertation are those that result in increased vulnerabilities in the network. The third question of interest is what type of attack graphs attack graph-based security metrics can evaluate. Some

attack graph-based security metrics have implicit requirements on the type of attack graphs they analyze.

When evaluating a network with these security metrics, a security engineer can use these attributes to avoid pitfalls and to improve network security. The process of improving network security is referred to as *network hardening.* Knowledge of attack graph-based security metrics' attributes can inform network hardening. We exemplify this proposition by demonstrating how attack graph-based security metrics can be used in network hardening to maximize security given a budget constraint.

1.1.2 Aggregation of Attack Graph-based Security Metrics

Although researchers have recognized that security is multidimensional [3, 5], proposed attack graph-based security metrics have been unidimensional. Because these security metrics can measure uniquely relevant aspects of network security, our approach combines their capabilities and respects the multidimensional nature of security. Using our characterization of attack graph-based security metrics, we specify an algorithm for determining between two network configurations which is most secure. We also demonstrate how multiple attack graph-based security metrics may be used with our network hardening approach.

1.1.3 Contributions of this Dissertation

The contributions of this dissertation are the following:

- Provide a characterization of attack graph-based security metrics that is based on the metrics' responses to security-relevant network events
- Propose a novel suite of attack graph-based security metrics
- Develop an algorithm for combining attack graph-based security metrics for evaluating two network configurations

- Design an algorithm for maximizing network security with respect to attack graph-based security metrics given a limited budget

1.2 Organization of this Dissertation

In chapter 2, we explain the various types of attack graphs and identify the attack graphs used in this dissertation. Chapter 3 reviews previously proposed network security metrics. In chapter 4, we propose our novel set of attack graph-based security metrics in this chapter. We also specify our algorithm for combining multiple security metrics to evaluate two network configurations in this chapter. In chapter 5, we present results from simulation studies that provide justification for characterizations of attack graph-based security metrics in chapter 3. We also present the results of our algorithm on synthetic attack graphs. In chapter 6, we specify our algorithm for performing network hardening given a limited budget. We conclude with our conclusions and future work in chapter 7.

2. THE ATTACK GRAPH

An *attack graph* is an abstraction that divulges the ways an attacker may use interdependencies among vulnerabilities to violate a security policy. The attack graph is derived from a network model description. This description includes extant vulnerabilities on hosts, host connectivity, and a security policy.

One question a security engineer contemplating the use attack graph analyses will confront is "what attack graph representation should be used?" This chapter provides the background needed to answer this question. Table 2.1 succinctly summarizes the analysis given below.

2.1 Related Work

In this section we give an overview of attack graph representations that have been reported in the literature. An attack graph is a collection of attack paths that are composed of conditions, exploits, or some combination of conditions and exploits. Based on these possible combinations, we classify the attack graphs into three broad categories: condition-oriented attack graphs, exploit-oriented attack graphs, and condition-exploit-oriented attack graphs. It is possible to convert one attack graph representation into another representation when all conditions and exploits are known.

2.1.1 Condition-oriented Attack Graph

In a condition-oriented attack graph, a node represents a subset of the network state, and an edge represents an exploit (or group of exploits) that moves the network from one state to another state. An exploit is a realized vulnerability. A

vulnerability specifies only its preconditions and consequences. An exploit details the specific network machines or software involved in realizing a vulnerability. A state is a network attribute or a set of network attributes. Network attributes include hosts, host connectivity, available software at hosts, access rights at hosts, and any other network characteristic deemed relevant to the modeler. State may be represented at various abstraction levels. Abstraction levels include the following: a single condition [6], a host [7], a host and privilege level [8], groups of hosts (e.g., subnetwork) [9], and the entire network [10]. Each abstraction level is represented by one or more predicates. In condition-oriented attack graphs predicates are used to describe vulnerabilities. That is, vulnerabilities are described by their preconditions and postconditions. When the preconditions for a vulnerability are satisfied, exploits cause more conditions (i.e., postconditions) to become true. These postconditions become available as preconditions for other vulnerabilities. We now examine specific instances of condition-oriented attack graphs.

Finite State Machine (FSM) Attack Graph

In a FSM attack graph (e.g., [10]), a node represents the state of the entire network. Transitions in the graph are made by attacker actions. Because this attack graph is an FSM, it avails itself to automatic generation with tools such as a model checker [10]. Although automatic generation is useful, the time to generate an attack graph for a practical network may be impractical [11]. This representation also suffers from redundant paths [12]. Moreover, the verbosity of the FSM representation makes human comprehension of the attack graph difficult [12].

Coordinated Attack Graph

A coordinated attack graph was proposed by Braynov and Jadliwala in [13]. A coordinated attack graph is an FSM attack graph where transitions into some states may require concurrent actions of multiple attackers. To represent this possibility, a

joint attack space is proposed. The joint attack space can represent the concurrent actions required to move from one state to another as well as the single attack actions that cause state transitions in the traditional FSM attack graph. The coordinated attack graph is unique in its ability to represent concurrent attacks. For instance, the coordinated attack graph can specify a symbolic link race condition vulnerability [13,14] involving two attackers. However, despite the flexibility of this representation, the coordinated attack graph suffers from the same disadvantages of the FSM attack graph.

Full Attack Graph

The full attack graph [7] shows all possible sequences of hosts and vulnerabilities an attacker can use from an entry point. In other words, the full attack graph shows all the paths an attacker can use to compromise hosts in a network. A host is represented by a node in the full attack graph. When generated, the full attack graph shows how the attacker can compromise any reachable host. Hence, the full attack graph is not inherently goal oriented. Thus, if an analyst wants to determine an attacker's ability to reach a target host, the analyst may prune or ignore attack paths that do not lead to the target host. The major drawback of the full attack graph is that it scales poorly [7]. Another drawback of this approach is that the full attack graph may have redundant paths [7].

Host-compromised Attack Graph

The host-compromised attack graph [7, 8] represents all hosts that an attacker may reach. This representation reveals what machines may be compromised without necessarily revealing the steps required to reach the hosts. The host-compromised attack graph acknowledges that an attacker may use a series of exploits to violate a security policy, but does not capture the exploits used to compromise hosts. A node represents a host in a host-compromised attack graph. Each edge in the graph

represents potentially many sequences of exploits that may be used to compromise a given host. An edge between two hosts represent the highest access level that the source host can obtain on the destination host via the exploit(s). A host-compromised attack graph is constrained to having hosts appear in the attack graph once. Like the full attack graph, the host-compromised attack graph is not a goal oriented approach. However, unlike the full attack graph, an analyst may not determine all the ways an attack may compromise a target host.

Predictive Attack Graph

In the predictive attack graph [7], a node represents a host and an edge represents a vulnerability. The predictive attack graph representation accurately forecasts the effect of removing vulnerabilities (i.e., edges) from the attack graph. The predictive attack graph is the full attack graph with redundant paths removed. A path is considered redundant if the path contains the same vulnerability-host pair in two or more places along the same attack path. This process of removing redundant paths, is referred to as "dynamic pruning." This process results in an attack graph that can be generated efficiently in practice [7]. A predictive attack graph may grow exponentially with respect to the number of hosts in the network. This growth may occur when firewalls cause an attacker's actions to take place in two steps: compromising hosts within a subnet directly, and then compromising other hosts in the same subnet that were not directly compromised through the firewall. This phenomenon is referred to as "firewall explosion." The predictive graph is not goal-oriented like the full attack graph. Moreover, there is no way of using only the predictive attack graph to determine all the attack paths an attacker may use to reach the goal state.

Node Predictive Attack Graph

In node predictive attack graph [7], a node can be host or a group of hosts, and an edge can be a vulnerability or a group of vulnerabilities. The node predictive

attack graph is a simplified version of the predictive attack graph. The node predictive attack graph's purpose is to mitigate the effects of "firewall explosion." Firewall explosion causes redundancy in the predictive graph. Thus, the node predictive attack graph mitigates this issue by merging nodes of the attack graph. Two nodes are merged if the attacker can compromise the two hosts from all hosts the attacker has already compromised. The generation of this graph has performed well in practice [7], however, Lippmann et al. admit that the generation process has many opportunities to "underperform." The method has many opportunities to underperform because the algorithm uses an optimistic approach for merging nodes together. Thus, in the worse case, all the merged nodes would have to be split. Because the node predictive attack graph is a simplification of the predictive graph, it too is not goal oriented.

2.1.2 Exploit-oriented Attack Graph

An exploit-oriented attack graph is the reverse of a condition-oriented graph with respect to nodes and edges. State is represented in the edges of the graph and the exploits are represented in nodes of the graph [12]. Exploit-oriented attack graphs may be referred to as exploit dependency graphs. A common representation of exploit-oriented attack graphs is to have unlabeled edges. The exploit-oriented attack graph's initial state(s) and the goal state(s) of the network are special nodes. Initial states are exploit nodes with null preconditions and true postconditions. Goal states are exploit nodes with true preconditions and null postconditions.

2.1.3 Condition-exploit-oriented Attack Graph

In a condition-exploit-oriented attack graph, state and exploits are represented by nodes [12]. An edge may relate a state and an exploit, or an exploit and a state. An edge may not relate a state and another state directly or relate an exploit and another exploit directly. When a state precedes an exploit in the attack graph, it

is considered a precondition for the exploit. When a state follows an exploit in the attack graph, it is considered a postcondition of the exploit.

Multiple Prerequisites Attack Graph

The multiple prerequisites attack graph is an attack graph where there are three types of nodes: states, prerequisites, and vulnerabilities [15]. State nodes represent the host and the access level obtained by the attacker. Prerequisite nodes represent the preconditions required for the attacker to realize a vulnerability, which is represented by a vulnerability node. The Multiple Prerequisite attack graph may be used with or without goal-orientation. This attack graph has been shown to have efficient run times in practice [15].

Logical Attack Graph

The logical attack graph [16] is a goal-oriented attack graph that has two types of nodes: fact nodes and derivative nodes. Also, there are two types of fact nodes: primitive fact nodes and derivative fact nodes. Primitive fact nodes have no preconditions and are unconditionally true (facts). Derivative fact nodes have preconditions. However, derivative fact nodes are not directly connected to primitive fact nodes. Derivative fact nodes connect directly to derivative nodes. Derivative nodes connect directly to primitive fact nodes. The set of primitive fact nodes making a derivative node true form a conjunction. The set of derivative nodes that make a derivative fact node true form a disjunction. Also, because edges represent the "depends on" relation, an edge appears between two nodes if the source node requires the destination node to be true in order for the source node to be realized. Thus, the source node "depends" on the destination node. The logical attack graph has been shown to have efficient run times for practical use [16].

Hybrid-oriented Attack Graph

The hybrid-oriented attack graph is described in [12]. A node represents a single condition. When multiple preconditions precede an exploit, the conjunction of these preconditions are required to realize the exploit. When state follows an exploit in the attack graph, it is considered a postcondition of the exploit. If an exploit provides multiple postconditions, it may provide any one postcondition; that is, exploits provide a disjunction of postconditions. This attack graph may or may not be used in a goal-oriented manner. The hybrid-oriented attack graph is expected to scale for hundreds of hosts [11].

2.2 Attack Graphs Used in this Work

In this dissertation we use condition-oriented attack graph, exploit-oriented attack graph, and the condition-exploit-oriented attack graphs. We use different attack graph representations based on what representation best helps convey our aim. If we do not explicitly state the representation being used, the representation will be clear from the context.

Table 2.1: Attack graphs and their respective disadvantages and advantages

Attack Graph	Drawbacks	Advantages
FSM	Verbosity, Impractical Generation Times	None
Coordinated	Verbosity, Impractical Generation Times	Can represent attacker collaboration
Full	Verbosity, Impractical Generation Times	The higher level of abstraction for nodes may enhance human comprehension
Host-compromised	Provides little information for network analysis	Practical generation times
Predictive	Suffers from "firewall explosion"	Practical generation times (generally)
Node-predictive	May underperform during attack graph generation, decreases information available for network analysis	Practical generation times and representation for humans
Multiple Prerequisites	May degenerate, requires worst case assumption	Practical generation times
Logical	Verbosity	Practical generation times
Hybrid-oriented	Verbosity, generation times for large networks	Generation times for moderately sized networks

3. NETWORK SECURITY METRICS

In this chapter, we begin by defining security metrics. We then detail classes of security metrics–identifying where attack graph-based security metrics reside within the classification scheme. We then review previously proposed network security metrics. In our review, we provide a detailed analysis of previously proposed attack graph-based security metrics. Table 3.1 shows how the reviewed network security metrics may be classified.

3.1 Security Metrics

The first word in "security metric" is security. *Security* is the process of providing confidentiality, integrity and availability to an entity according to some policy. With respect to an organization, security may be split into four domains: physical security, personnel security, information technology (IT) security, and operational security [17]. *Physical security* refers to providing protection for hardware, software, and information against physical threats. *Personnel security* refers to the policies and procedures used to ensure that an organization's staff is properly prescreened, trained, and later monitored. *IT security* refers to the technical features and functionality that contribute to an IT infrastructure's security. *Operational security* refers to the policies and procedures that are enacted to govern how and when users can securely interact with systems and system resources.

The second word in "security metric" is metric. A *metric* is a value that facilitates decision making and is derived from measurement. Metrics measure the attributes of entities [18, 19]. Generally, there are two types of attributes: internal and external. *Internal attributes* are characteristics that are inherent to the entity itself. *External attributes* are the actions performed by the entity while in operation in some envi-

ronment. For instance, a network can be considered an entity. The existence of a network-based intrusion detection system (NIDS) in the network can be considered an internal attribute. The NIDS's performance on real network traffic can be considered an external attribute.

Deriving inspiration from the Systems Security Engineering Capability Maturity Model [20] and Herrmann in [21], we define *security metrics* as values produced from measuring identifiable entity attributes that effect physical, personnel, IT, or operational security. Security metrics can be classified into the following categories: Return on investment (ROI) metrics, Resiliency metrics, and Compliance metrics [17]. *ROI metrics* measure an organization's monetary benefit from dedicating resources toward security controls. *Resiliency metrics* measure an organization's ability to maintain acceptable service in the presence of attacks or failures. *Compliance metrics* measure how well an organization conforms to existing regulations and standards.

In this dissertation, we are concerned with network security metrics. By focusing on attack graph-based security metrics, we have focused our contribution to security metrics that measure the internal attributes of networks.

3.2 Classes of Network Security Metrics

In this section, we detail four classes of network security metrics we have developed based on the literature. There are two primary classes and three secondary classes. The two primary classes of network security metrics are architectural-based metrics and performance-based metrics. The three secondary classes of network security metrics are time-based metrics, probability-based metrics, and complexity-based metrics. All metrics belong to a primary class but not necessarily a secondary class. In describing each security metric class, we also detail associated drawbacks.

3.2.1 Primary Security Metric Classes

The primary security metric classes are architectural-based security metrics and performance-based security metrics. The difference in the two classes stems from the type of attributes they measure. Architectural-based metrics measure internal attributes. Performance-based metrics measures external attributes.

Architectural-based

Architectural-based network security metrics measure the internal attributes of a network. Internal attributes may include, for example, services available on a network, connectivity of hosts on the network, or extant vulnerabilities. If a metric is measuring an attribute of a network that is not a behavior or action, then the metric is measuring an internal attribute. Such measuring may be done through a static analysis of the entity.

The primary drawback of architectural-based network security metrics is that they do not measure external attributes. External attributes give the most accurate view of how a network will perform in a real-world environment. When measuring internal attributes, inferences must be made about how the network will perform. If a false inference is made, then conclusions drawn about a network's security may be erroneous. Although inferences must be made when measuring external attributes, the inferences are dependent upon whether the measured entity will perform the same in the future. Because the entity is measured on a specific data set, the inference is usually narrowed to "the entity will perform the same in the future on the same data set." For instance, a data set could be comprised of various forms of malicious software. Thus, if a security engineer has malicious software that accurately models what the network will encounter in the future, the security engineer can be fairly confident that the entity will react consistently in the future with prior evaluations. Whether this narrowed inference is useful depends on what is being measured and the

goals of the security engineer. In general, identifying data sets that allow for more general claims remains an open problem.

Attack Graph-based

Attack graph-based security metrics is a type of architectural metric. An attack graph-based security metric is a value produced from measuring the internal attributes of a network that affect IT security or operational security. The values are derived from generating an attack graph and subsequently deploying an analysis over the attack graph. This analysis is the measurement that produces the attack graph-based security metric.

Although attack graph-based security metrics may be interpreted as a compliance metric [21], we believe it is more useful and accurate to classify them as "Non-compliance metrics." In general, non-compliance metrics specify to what degree an entity is non-compliant with respect to some security policy. Measuring non-compliance provides in-depth perspective of security policy violations. This perspective allows for addressing security issues at a finer granular level.

The reasons for non-compliance are many. With respect to attack graphs, non-compliance exists because vulnerabilities persist in organizations. Given the primary objectives of an organization (e.g., profitability), a security engineer may be unable to remove certain vulnerabilities. This inability may be the result of a lack of resources to remove the vulnerabilities. Alternatively, offered solutions that fix the vulnerabilities may be deemed unsuitable. This situation may arise when before a patch is run on production level systems, an organization requires the patch pass certain tests to ensure the patch does not adversely effect other parts of the system. If the patch does not pass these requisite tests, it will be unable to run on the production level systems. In the case where the patch does not pass, the production level systems would have to continue running with the known vulnerabilities. Another scenario where a solution for removing vulnerabilities may be deemed unsuitable is when the solution requires coarse action (e.g., shutting down ports) that the organization is unable to take

because of its primary organizational objectives. Aside from offered solutions that are inviable, a reason for persistent vulnerabilities may be no available solution for the vulnerability at the time of discovery. Thus, until a solution is released, the organization may need to run its systems with known vulnerabilities.

When a system is not 100% compliant for a given set of security policies, a non-compliant metric is applicable. For instance, a security policy may state that "an unauthorized user should not gain root privilege on an administrator machine." If this security policy can be violated through the network, then the attack graph-based security metric, the Number of Paths metric, can be used to identify the number of ways an attacker can violate this policy. This examination of how egregious non-compliance is in an organization improves the granularity at which security degradations or improvements can be assessed. Continuing the previous example, if the initial number of attack paths is 8, and after instrumenting some countermeasures the Number of Paths metric goes down to 3, then one would have evidence that the security controls put in place were effective at improving security. However, at the level of the security policy, whether the number of attack paths was 8 or 3, the security policy is violated with no distinctions between the two networks. Because no system can be 100% secure with any certainty, the development of non-compliance security metrics appears to be a promising direction.

When an attack graph-based security metric classifies one network as more secure than another network, it is actually stating that the more secure network is *less non-compliant* with the observed security policy. Full security policy conformance suggests that a network is secure with respect to the given security policy. An attack graph-based security metric does not suggest this about a security policy. An attack graph-based security metric can specify only that a network is non-compliant and that one network is more or less non-compliant than another network. An attack graph-based security metric makes no assertions about how compliant a network is with a security policy unless the security policy is designed for the attack graph (e.g., "no attack paths should be generated by the attack graph for any network

security policy"). This limitation exists because attack graph-based security metrics are capable of partially addressing only two of the four security domains: IT security and operational security.

IT security deals with all aspects of the IT infrastructure of an organization. Everything from the network media being used to the software running on the network hosts is considered a part of an organization's IT infrastructure. Because an attack graph is based on the description of the network model, a security metric that measures the attack graph provides some insight into the security of the IT infrastructure underlying the attack graph. Operational security deals with all the ways users may interact with the system. Because an attack graph represents a chain of exploits an attacker can leverage to violate a security policy, a security metric based on the attack graph gives some insight into how a user (authorized or unauthorized) may interact with a network in insecure ways which is relevant to operational security.

Because attack graphs focus *exclusively* on vulnerabilities in the network which may or may not be the result of inappropriate use of software and hardware, personnel and physical security is not captured by the attack graph. An interesting area of research may be determining how the attack graph can be repurposed for these other two domains of security.

While attack graph-based security metrics attempt to measure IT security and operational security, they do so only partially. There are three primary reasons for this shortcoming.

- *There is no guarantee that the description of the network model is completely correct.* The level of detail that a modeler can use is dependent on the attack graph generator being used and the modeler's knowledge of the system being assessed. The most pervasive issue is that the modeler can never be assured that *all* vulnerabilities are accounted for when modeling the system. Even when the attack graph generator provides sufficient flexibility and the modeler has enough knowledge about the network to model it accurately, issues may still arise in the modeling of vulnerabilities. Since vulnerabilities are specified

in terms of their preconditions and postconditions in an attack graph, a modeler must know the exact conditions that allow the vulnerabilities to be realized and their exact consequences. Obtaining the exact conditions for vulnerabilities is difficult in general, especially if the vulnerabilities are not listed at an online resource such as MITRE CVE [22]. Even if the vulnerability is found, human error in transcribing discovered vulnerabilities that are written in vague terms to a form amenable for an attack graph generator could produce faulty network models that would subsequently produce faulty analyses.

- *IT and operational security include the processes that were used to build and maintain a network and attack graphs do not.* The attack graph does not take into account the processes that were used to develop the network resources. Systems which undergo the Common Criteria [23] are believed to be more secure than systems that do not adhere to the Common Criteria development process. If patches that are added to a network undergo rigorous testing and code review before being instrumented into the network, then this network would be believed to provide higher assurance than a network that does not perform any structured testing of new patches applied to the network.

- *Attack graph-based security metrics cannot assess the performance of security controls.* Attack graph-based security metrics measure internal attributes of a network. The number of hosts that are involved in some attack scenario is an example of an internal attribute. The performance of security controls are external attributes of a network. To truly assess the performance of security controls, a dynamic environment is required and attack graph-based security metrics are static analyses.

Performance-based

Performance-based network security metrics measure a network's external attributes. Human performance may be included as part of the external attributes of a

network. This assertion is intuitive as a network typically requires human intervention to operate properly on a continuing basis. An issue that pervades performance-based metrics is choosing an appropriate data set to evaluate an external attribute. Inferences made about specific data sets may only be marginally useful. For instance, if an intrusion detection system (IDS) demonstrates that it has the ability to prevent a specific attack, then this is all that can be inferred. If the attack changes form in some way, the IDS may or may not be able to detect the attack. The ability to come up with variant forms of an attack is a function of the resourcefulness and creativity of the security engineer evaluating the IDS.

3.2.2 Secondary Security Metric Classes

The secondary security metric classes are probability-based security metrics, complexity-based security metrics and time-based security metrics. These metrics can be applied to internal and external attributes of a network.

Probability-based & Complexity-based

Probability-based network security metrics use probabilities to arrive at results. Probabilities may be the likelihood of a network being attacked, an attacker choosing an action, an attacker successfully violating a security policy, or an attacker exploiting specific vulnerabilities. Such probabilities may take into account the difficulty associated with exploiting a vulnerability. This possibility is why we discuss probability and complexity together. Moreover, there is much overlap in the drawbacks of probability-based and complexity-based network security metrics. The drawbacks of these type of metrics can be classified into historical data, assumptions, expert opinion, and qualitative values.

Historical Data

Probability values are usually obtained from historical data. Historical data really

refers to *detected* security incidents from the past. The sole usage of recorded historical data indicates that there could be a relevant number of attacks that occur, that go undetected, and do not factor into probability estimates. One has no assurance that the sample set of detected security incidents infers the set of all security incidents. Furthermore, if a company has little to no historical data, probabilistic approaches would be unavailable to them. A probabilistic approach is difficult to implement when a security metric requires probabilities to be assigned individually. In order to accomplish this task, appropriate investments would need to be made to ensure precise finely tuned audit data is created, processed and pursued. These activities are not performed by many organizations [24].

When complexity values are being assigned, one must ensure that historical values correspond to present day values. For instance, if a vulnerability was generally accepted to be extremely difficult to exploit three months ago, this perspective may no longer be true today. The change in difficulty associated with exploiting the vulnerability may have decreased because new knowledge has become publicly available regarding the vulnerability (e.g., the release of a script that allows novices to exploit the vulnerability). Having a consistent representation of what potential attackers know about vulnerabilities is an arduous task.

Assumptions

A common method assumes that vulnerabilities are independent although these vulnerabilities exists within a dependency graph where previous exploits may decrease the difficulty of exploiting later attacks [25]. For instance, when an attacker manages to read a "shadow" file on a Unix-based system, the attacker may be able to use these hashed value passwords to assist in compromising other hosts on the network.

Another common assumption is that an attacker's success probability can be captured by a single value. Since attackers vary in tenacity, motivation, skill, and money, these probabilities can be vastly different for each vulnerability [26]. Hence, multiple probabilities would be needed to represent the probability of attack of a specific

vulnerability which is an arduous task. An issue which supersedes the above problem is that quantifying things such as tenacity, motivation, and skill can be just as challenging as assigning probabilities to vulnerabilities.

Expert Opinion

Another approach for obtaining probabilities or complexity values is through expert opinion. It is unclear to the community what constitutes a qualified expert in the domain of recommending probabilities or complexity values for vulnerabilities or attack actions. What experiences should an expert have in order to predict the success of an attacker compromising a set vulnerabilities? Even if a "white hat" (an ethical hacker), has experience with hacking, there is no guarantee that the hacker has experience with hacking into a network system similar to the one of interest. Moreover, any probability/complexity estimates the hacker could provide would be consistent with the hacker own beliefs not statistically valid probabilities. In referring to the Annual Loss Expectancy approach, which relies heavily on knowing the probability of a particularly threat becoming realized, Schneier calls the approach "a lot of guesswork," and Schneier would likely be considered an expert by many [27]. Others have noted the impracticality of obtaining reliable probability estimates [2].

The issue of an expert pontificating from a single frame of reference applies equally to the process of assigning complexity values to vulnerabilities. What is difficult for one attacker, can be easy for another attacker. The difficulty in assigning complexity values to attacks mirrors the difficulty of assigning probabilities to vulnerabilities. Generally, the amount of data needed to make such a determination is impractical.

Experts are not immune to the cognitive pitfalls that are associated with making assessments [28]. The pitfalls include: availability heuristic, anchoring heuristic, framing effects, blind obedience, and premature closure. The examples explaining the above pitfalls, are given within the context of the medical field in [28]; however, they can be extended for the security field. The availability heuristic causes an individual to come to a conclusion based on the ease of recalling past events. If trying to assign

a likelihood to a vulnerability, the expert may assign the vulnerability a probability value based on the number of times he has seen the vulnerability on online forums or in MITRE CVE [22], Bugtraq [29], or CERT [30]. The anchoring heuristic relies on initial impressions to make decisions. An expert may assign an attack to be the easiest attack, and then modify his viewpoints about other attacks to ensure the initially chosen attack is the "easiest" attack. Framing effects is having a decision affected by subtle wording. If an expert reads that a new zero-day vulnerability caused an "annoyance" versus a "catastrophe" for an organization, the expert may be more prone to underestimate the potential impact of such a vulnerability. Blind obedience is the act of giving undue deference to authority or technology. An example would be an expert that defers completely to Common Vulnerabilities Scoring System (CVSS) [31]. Premature closure is the act of focusing on a single idea without pursuing alternative ideas. An example would be an expert that assigns success probabilities to vulnerabilities without consulting with alternative sources (e.g., CVSS). With the problems associated with assigning complexity or probability values, using expert opinion to secure may produce misleading results.

Qualitative Values

In [32], IEEE defines failure as "the inability of a system or component to perform its required functions within specified performance requirements." A security failure is a failure that negatively effects confidentiality, integrity or availability. In [33], the International Electrotechnical Commission (IEC) stipulates quantitative likelihoods should be used with random failures, whereas qualitative likelihoods should be used with systematic failures. The security failures that arise from a series of vulnerability exploits in an attack graph are decidedly systematic. This observation would suggest that using success probabilities in attack graph-based security metrics are actually qualitative likelihoods and not quantitative likelihoods placing such values onto an ordinal scale. No arithmetic or algebra could be performed on these values as this

would be mathematically unsound. This assertion holds for qualitative complexity values as well.

Despite the difficulties of deriving complexity, we attempt to provide approaches reasoning about the difficulty associated with attack paths (see section 4.1). Having a systematic way of approaching attack path complexity would provide consistency in how complexity values are assigned to attack paths.

Time-based

Time-based network security metrics yield time values as their result. Time-based network security metrics are used to measure how quickly an network or organization can be penetrated or how quickly an network or organization can respond to attacks or proactively preempt attacks. These metrics may also include measuring changes in internal attributes (e.g., TVM see section 3.3.10). When the measuring process involves assessing the actions of humans, the repeatability and reproducibility of experiments becomes more difficult.

3.3 Related Work

In this section we describe network security metrics that have been proposed in the literature. Although the contribution of this dissertation deals with attack graph-based security metrics, we review other types of network security metrics to provide context. At the end of this section, we provide a table that classifies each metric discussed in this section.

3.3.1 Security Vulnerability Index (SVI) Metric

Alves-Foss and Barbosa [34] propose the use of a System Vulnerability Index (SVI). The SVI value represents the system's vulnerability to common attacks on a scale from 0 to 1. A higher SVI represents higher vulnerability. A value of 0 to .15

indicates low vulnerability. A value between 0.15 and 0.3 equates to a moderate level of vulnerability. A value of 0.3 to 0.6 suggests the system is vulnerable to common attacks. A value greater than 0.6 is classified as extremely vulnerable. SVI is composed of rules that belong to one of the following 3 categories: system characteristics, potentially neglectful acts, and potentially malevolent acts. Rules takes the form **if** *"antecedent"* **then** *"consequent"*. The *"antecedent"* contains the conditions required to obtain the value given in the *"consequent"*. Conditions indicating a high vulnerability in the antecedent solicit high certainty indices (CIs) in the consequent. A certainty index is the degree of belief in a hypothesis and is a value between 0 and 1. Once all rules have been developed to describe the three aforementioned vulnerability categories, the SVI is calculated as:

$$\text{SVI} = \text{CI}[\text{h}, r_i \text{ AND } r_j] = \text{CI}[\text{h}, r_i] + \text{CI}[\text{h}, r_j] \times (1 - \text{CI}[\text{h}, r_i]).$$

The above formula is to be read as the "measure of belief in hypothesis h given that rules r_i and r_j are true is equal to the degree of certainty from the first rule r_i, summed with the degree of certainty from the second rule, r_j, scaled by the lack of supporting evidence from the first rule." Combining more than two rules involves using values already computed as a single value. This new single value would take the place of r_i in the above formula.

SVI is an architectural-based metric and a complexity-based metric. The approach involves the static assessment of existing system components. This static assessment makes SVI an architectural-based metric. Because hypotheses regarding the vulnerability of system components must be made, SVI is also a complexity-based metric.

3.3.2 Mean-Time-To-Breach (MTTB) Metric

Jonsson and Olovsson use Mean-Time-To-Breach (MTTB) as a security metric in [35]. Hours are used as the time unit. Let t_r be the total amount of time the red team r spent attempting to breach a system. t_r is comprised of preparation time and

attack time. Let $NumB_{t_r}$ be the total number of breaches the red team r was able to successfully accomplish over t_r. Then MTTB is given by the following equation:

$$MTTB := \frac{t_r}{NumB_{t_r}}$$

This metric implicitly favors the quantity of breaches over the quality of breaches. For instance, a successful attack could cause a single breach that is more severe (e.g., provides root-level access) than ten other breaches (e.g., provides user-level access), and this metric considers the former scenario more secure. McDermott in [36] notes that little information can be gleaned from the MTTB metric. A CSO examining a system's MTTB is given little information as to how to improve the system based on the MTTB metric. Any information obtained would be from the red team's documentation.

MTTB is a performance-based metric and a time-based metric. Since this metric measures human performance, this metric is performance-based. Also, because this metric measures human performance with respect to time, this metric is also time-based.

3.3.3 Work Factor (WF) Metric

Work factor [37] is a security metric proposed by Schudel and Wood. This metric is based on the hypothesis that adversary work factor is quantifiable and yields the relative strengths and weaknesses of complex information systems. Work factor is given by $t_{r_{sec}}$, where $t_{r_{sec}}$ is the amount of time the red team r needed to compromise the system by violating security policy *sec*. This distinction gives the CSO more information than the MTTB metric because it specifies the security policy being violated. However, this technique does not capture cases where the red team may exploit the same vulnerability different ways. This information would be expected to be found in the red team's documentation.

WF is a performance-based metric and a time-based metric. WF measures human performance using time units. Therefore, WF is both performance-based and time-based.

3.3.4 Mean-Time-To-Recovery Metric

Mean time to Recovery (MTTR) refers to the average amount of time an ally requires, in the event of an attack, to bring a system out of a compromised state to a state that is not compromised [2]. Faster MTTRs correspond to more favorable security. If this metric is applied to an intrusion-tolerant system [38], it will measure how quickly the system's software and/or hardware can recover from being compromised. In the absence of intrusion-tolerant systems, the MTTR is based primarily on the ability of the system's stakeholders to restore the system to an uncompromised state. The uncompromised state may be some alternate operable state and not necessarily the original uncompromised state the system was in before being moved into the compromised state. A definition for MTTR is given in [39]. Let T represent the total recovery time for each incident over some period of time. Let R represent the total number of repairs done over this time period. Then MTTR is given by the equation below.

$MTTR = \frac{T}{R}$

MTTR is a performance-based metric and a time-based metric. MTTR measures how quickly a system can be transitioned from a compromised state to a safe state. Therefore, this metric is both performance-based and time-based.

3.3.5 Mean-Time-to-First-Failure (MTFF) Metric and Mean-Time-To-Failure (MTTF) Metric

Sallhammar et al. [40] propose the use of Markov Chain analysis to predict a system's security level over time. Two measures used include: the Mean Time to First Failure (MTFF) and the Mean Time To Failure (MTTF) metrics. MTFF corresponds

to the first time the system enters a failed state from the point of system initialization. MTTF corresponds to the first time the system enters a failed state from some randomly chosen uncompromised state. The system is given as a state transition model. Events trigger state transitions. These events correspond to attack actions that move the system into a failed state. An attacker may have a number of actions available in a given state. Each action from a state has an associated probability of transitioning into every other state. These probabilities are obtained using a two-player zero-sum game theoretic approach. This approach also requires choosing accumulated attack intensities which predicts failure over some time period of time t.

There are states S_1 through S_n in the system. Let S_G correspond to good states S_1 through S_k. Let S_F correspond to failed states S_{k+1} through S_n. Let Q be an $n \times n$ state transition matrix. Split the matrix into four quadrants such that Q_1 is a $k \times k$ matrix, Q_2 is a $k \times n-k$ matrix, Q_3 is a $n-k \times k$ matrix, and Q_4 is a $n-k \times n-k$ matrix. Let $T = \{T_1, T_2, ...T_k\}$. Once the following equation is solved, $-TQ_1 = \{1, 0, ..., 0\}$ MTFF is given by:

$MTFF = \sum_i^k T_i$

The equation for MTTF is given below.

$MTTF = \frac{X_G(-Q_1)^{-1}h_k}{X_G h_k}$

X_G corresponds to steady-state probabilities for states in S_G. h_k is vector of k ones.

MTFF and MTTF are architectural-based metrics and probability-based metrics. Also, these metrics require that probabilities for states be discovered. This requirement makes these metrics probability-based. Identification of appropriate states require examining internal attributes of the system. This property makes these metrics architectural-based metrics.

3.3.6 Mean-Time-Between-Failures (MTBF) Metric, Mean-Time-Between-Maintenance-Actions (MTBMA) Metric, Mean Downtime (MDT) Metric, and Mean Maintenance Time (MMT) Metric

The Mean-Time-Between-Failures (MTBF) metric differs from the MTTF metric in that MTBF refers to components that can be fixed–whereas MTTF may or may not refer to fixable components. The Mean-Time-Between-Maintenance-Actions (MTBMA) metric measures the time used to perform corrective and preventive maintenance on the network. The Mean Downtime (MDT) metric measures the average downtime for a network over some period of time. The Mean-Maintenance-Time (MMT) metric refers to the average amount of time required to perform maintenance actions. The equations for these metrics follow from [21].

$MTBF = T/F$, where T is the duration of the period of interest, and F is the number of failures experienced over T.

$MTBMA = T/M$, where T is the duration of the period of interest, and M is the number of maintenance actions performed over T.

$MDT = D_T/N_T$, where D_T is the sum of all the downtime experienced over T, and N_T is the number of times downtime occurred over T.

$MMT = \frac{C_{1000} \times MTTR_C + P_{1000} \times MTTR_P}{C+P}$, where C_{1000} refers to the number of corrective maintenance actions performed per 1000 hours, $MTTR_C$ refers to the MTTR for corrective maintenance, P_{1000} refers to the number of preventive maintenance actions performed per 1000 hours, $MTTR_P$ refers to the MTTR for preventive maintenance, C refers to the number of corrective maintenance actions carried out, and P refers to the number of preventive maintenance actions taken.

From these metrics, we may obtain other metrics [39, 41].

Traditional availability [21] is given by [41]:

$A_{traditional} = MTBF/(MTBF + MTTR)$

Operational availability, which is the availability observed after system deployment [39] is given by:

$A_{operational} = MTBMA/(MTBMA + MDT)$

Achieved availability which refers to the observed availability once the system reaches steady state is given by [39]:

$A_{achieved} = MTBMA/(MTBMA + MMT)$

Each of the abovementioned metrics are performance-based metrics and time-based metrics. This classification is due to these metrics measuring human performance over some duration.

3.3.7 False Positive Rate (FPR) Metric and False Negative Rate (FNR) Metric

False positive rates (FPR) and false negative rates (FNR) are used to assess intrusion detection systems and firewalls. An IDS produces a false positive when it classifies network activity as an attack when the activity is not an attack. A firewall produces a false positive when it classifies benign network traffic as malicious. An IDS produces a false negative when it is fails to detect an attack. A firewall produces a false negative when it fails to block connectivity from an unauthorized address and or port. The quality of these measures is directly related to the quality of the data set used to evaluate the IDS or firewall.

Let D be the data set used to evaluate an IDS or firewall. Let $B \subseteq D$ be data items that should be detected as malicious by an IDS or should be blocked by a firewall. Let X be the elements that IDS detected as malicious or the elements that a firewall blocked. Let $X_1 \subseteq X$ such that each $x \in X_1$ is also $x \in D - B$ Then, the false positive rate is given by:

$FPR = \frac{|X_1|}{|D-B|}$

Let $X_2 \subseteq X$ such that each $x \in X_2$ is also $x \in B$. Then the false negative rate is given by:

$FNR = 1 - \frac{|X_2|}{|B|}$

Both metrics are performance-based metrics. Since these metrics measure how a security control behaves on a data set, the external attributes of these security controls are being assessed. Their classification stems from this assessment of external attributes.

3.3.8 Accumulated Coupling Coefficient (ACC) Metric

To develop a framework for predicting and mitigating software attackability in the early stages of the development process, Liu et al. [42] studied the empirical relationship between design and attackability. In this framework, the software services are represented using the User-System Interaction Effect model (USIE). A USIE model is a graph $U = (N, E, <)$, where N is a set of nodes, E is a set of edges and $<$ is a partial order relation over E. The USIE model involves two kinds of nodes namely *InteractionStart* and *RoleEntity* nodes. *InteractionStart* nodes represent the starting-point. *RoleEntity* represent the role entity of a user-system interaction. A RoleEntity node is identified by a name and has a set of security characteristics. Edges in a USIE model are directed from a source role entity to a target role entity. The edges are accompanied with attributes that describe the privileges underlying the communication. For example, a basic set of attributes consists of {*read, write, execute, create, delete*}. Given two software services c_i and c_j their coupling coefficient $CoupCoef(c_i, c_j)$ can be computed by counting the number of shared persistent entities between the two services, which can be derived from their USIE models. Liu et al. [42] define the coupling coefficient metric as follows:

Given two USIE models $U_1 = (N_1, E_1, <)$ and $U_2 = (N_2, E_2, <)$, the number of shared persistent entities between U_1 and U_2 is defined as:

$$CoupCoef(U_1, U_2) = \sum_{n_i \in N_1 \cap N_2} p(n_i)$$

$$\text{where } p(n_i) = \begin{cases} 1 & \text{, if } n_i \text{ is a persistent RoleEntity node.} \\ 0 & \text{, if } n_i \text{ is a transient RoleEntity node.} \end{cases}$$

Furthermore, Liu et al. [42] provided the accumulated coupling coefficient to measure the coupling between two sets of services. Given two service sets C_i and C_j, the accumulated coupling coefficient is between the two sets C_i and C_j is defined as follows:

$$AccCoupCoef(C_i, C_j) = \frac{\sum_{c_k \in C_i} \sum_{c_l \in C_j} CoupCoef(c_k, c_l)}{|C_i| \times |C_j|}$$

By comparing the attackability of systems with different coupling coefficient values, Liu et al. [42] concluded that the attackability of a system increases as the coupling increases for its software services. This correlation has been shown for Denial of Service (DoS) attacks.

ACC is an architectural-based metric, because it assesses system components, and these components do not need to be in operation to compute ACC.

3.3.9 Langweg (LW) Metric

Langweg [43] proposes a metric for determining the resistance of an application to malware attacks. Since it is infeasible to prove that a system is 100 percent secure, Langweg posits that there must exist a spectrum of secureness between completely insecure and completely secure. The security requirements of an application *A1* define what degree of security is acceptable for *A1*. Without understanding the standards of security applications must adhere to, the comparison of two applications can be nonsensical if the applications have different standards of security. To deal with the issue of varying standards of security, Langweg proposes a classification scheme that provides a partial ordering on security requirements. Security requirements are ranked along three dimensions. The dimensions relate to the degree the requirements mandate the limitation of damage on code integrity, data integrity, and data confidentiality. Similarly, an attacker's capability is measured along four dimensions: the ability to initiate an attack, the ability to carry out attack over time, the ability to customize the attack while it is in progress, and the ability to influence the user. Henceforth, a system *Sys1* is more secure than another system *Sys2* if *Sys1*'s security

requirement $sr_{Sys1} \geq sr_{Sys2}$ and *Sys1* can successfully resist an attacker of attack capability $ac_{Sys1} \geq ac_{Sys2}$. Thus, if a higher attack capability level is required to breach *Sys1* then *Sys1* is more secure than *Sys2*. The assumption is that weaker security requirements suggests a weaker system.

Because LW does not require any dynamic analysis to obtain its value, this metric is an architectural metric. All that is required is a static analysis.

3.3.10 Total Vulnerabilities Measure (TVM) Metric

In [44], Abedin et al., propose a metric called the Total Vulnerability Measure (TVM). TVM is the sum of two other proposed metrics called the Existing Vulnerabilities Measure (EVM) and the Aggregated Historical Vulnerability Measure (AHVM). The equation is given as

$TVM(A) = NP_1 EVM(A) + NP_2 AHVM(A)$.

NP_1 and NP_2 represent weights a user would want to give the two vulnerability measures. EVM is given by the following equation:

$EVM(A) = NCP_1 \ln \sum_{v_i \in EVM_s(A)} e^{SS(v_i)} + NCP_2 \ln \sum_{v_i \in EVM_u(A)} e^{SS(v_i)}$.

$EVM_u(A)$ corresponds to vulnerabilities that are not patched in system A. $EVM_s(A)$ corresponds to vulnerabilities that have known solutions in system A. $SS(v_i)$ is the Severity Score (SS) assigned to vulnerability v_i. The higher a vulnerability's severity score is, the more severe the vulnerability is to a system.

The authors use exponential averages (instead of arithmetic or geometric averages) because they wanted to ensure that average scores are at least as high as the highest vulnerability score in the set. The authors suggest that two factors (Exposure and Traffic Rate) be taken into account when considering the severity a vulnerability may have on a service S. These factors should be multiplied with the Severity Score determined for the vulnerability. One factor is the Exposure Factor (EF). Its equation is given as follows,

$EF(S) = 1 + \frac{\log_2(IP(S)PORTS(S))}{\log_2(2^{32}2^{16})}$.

The numerator corresponds to the IP addresses and ports that S services. The denominator refers to the total IP address and port number space. The other relevant factor is the Traffic Rate Factor (TRF). Its equation is given as follows,

$TRF(S) = 1 + \frac{\text{Traffic volume of S}}{\text{Total traffic volume}}$.

Another metric proposed in a related paper [45] is the Historical Vulnerabilities Measure (HVM). Its equation is given as the following:

$HVM(S) = \ln(1 + \sum_{X \in H,M,L} w_X \sum_{v_i \in HV_x(S)} DV(v_i))$.

The set of vulnerabilities for a service are determined and divided into groups based on the following risk levels: High, Medium, and Low. HVM is modeled as a exponential decay function. The equation for the decay function is given by the following:

$DV(v_i) = SS(v_i)e^{-\beta Age(v_i)}$

The Aggregated Historical Vulnerabilities Measure (AHVM) the average of all services' HVM score in a system. This is given by the following equation:

$AHVM(A) = \ln(\sum_{s_i \in SERVICES(A)} e^{HVM(s_i)})$

Through setting parameters of the HVM equation appropriately and looking at services listed in National Vulnerabilities Database [46], the authors obtained results that suggests that a service S_1 that has a higher HVM score than another service S_2 will have more vulnerabilities than S_2 in the next time period (6 months to 2 years) being examined. The authors in [45] show that with appropriate parameter tuning of HVM that an prediction accuracy up to 83 percent with NVD may be obtained.

TVM is architectural metric. It requires the static assessment of the network being measured.

3.3.11 Probabilistic Vulnerability Measure Metric and Attack Propagation Metric

In [45], Ahmed et al. propose two metrics: a Probabilistic Vulnerability Measure (PVM) and Attack Propagation Metric (AP). PVM is used for determining how

likely a vulnerability will be released for a service over some time period and the vulnerability's expected severity. The equation is given by the following

$PVM(S) = \ln \sum_{s_i \in S} e^{ER(s_i)}$

$ER(s_i)$ is the Expected Risk (ER) of a service s_i. ER is given by the following equation,

$ER(s_i) = P_{s_i} E[X_{s_i}]$

P_{s_i} is the probability service s_i will experience a vulnerability over some time period T. $E[X_{s_i}]$ is the expected severity a vulnerability will have on a service s_i. The authors used an exponential distribution and an empirical distribution to attempt to determine the appropriate distribution for P_{s_i}. Empirical results suggested that the exponential distribution provided more accurate results.

AP attempts to capture the damage that may result from any host in the network that the attacker can reach. The AP metric is given by the following equation,

$AP(D) = \sum_{d \in D} P(d) SBE_d$.

D represent all hosts that may be entry points for an attacker. $P(d)$ is the probability that a vulnerability exists on host d. SBE_d represents the Service Breach Effect (SBE) for an entry point node d. This value is derived from a Service Connectivity Graph (SCG) that connects hosts to other hosts via services. Hosts are represented by nodes. Edges represent the different services that connect one host to another host. The equation for SBE is given by the following,

$SBE_d = \sum_{n \in N} (\prod_{m \in nodes\ from\ d\ to\ m} p_{s_{dm}}) Cost_n$

The authors do not provide an explicit definition for p_s, but state that p_s represents a decreasing function of EVM and HVM. SBE represents the effect of a host d reaching every other reachable host. The effect, or damage, is captured in $Cost_n$.

PVM and AP are architectural-based metrics and probability-based metrics because they require static analysis of the network being measured and the assignment of probability values.

3.3.12 Attack Surface (AS) Metric

The idea of an attack surface was first proposed by Michael Howard in [47]. The idea was formalized by Howard et al. in [48]. That version of the attack surface formulation was composed of three dimensions: targets and enablers, channels and protocols, and access rights. Each dimension represents a different type of resource found on a system. The calculation is performed by summing up all instances (with their corresponding weights) of relevant resources found in the system resulting in a single value for the system attack surface measure.

An issue with this approach is that there is no systematic way for assigning weights to various resource instances. Another drawback was that there was no systematic way for identifying the relevant resource instances. Lastly, there was no justification in adding together calculations obtained across dimensions.

The aforementioned issues are addressed by Manadhata and Wing in [49,50]. The dimensions are changed to: methods, data, and channels. Methods refer to the API available to the environment. Data refers to persistent data that can be accessed by the environment. Channel refers to the communication channels that can be used to access methods and data.

A total ordering is imposed on access right levels, privilege levels, types of data and open channels. Higher values are assigned to levels/types that could cause more damage. For example, a process, *p1*, with root privilege could cause more damage than another process, *p2*, with non-root privileges, and therefore, *p1* would be assigned a higher numerical value than *p2*. This scheme addresses the issue of assigning weights to resource instances. Relevant resource instances for all dimensions are identified by Manadhata and Wing's entry point and exit point framework. Entry point analysis refers to the ways that data may flow into a system from the environment. Exit point analysis refers to the ways that data flows out of the system to the environment. Since there is no justification for adding values across dimensions, the measures obtained

are not added across dimensions. This change in calculation yields a 3 dimensional value. The calculation for attackability of a system is given by [51]:

$(Method = \sum_{m \in M} der_m(m),$

$Data = \sum_{d \in I} der_d(d),$

$Channel = \sum_{c \in C} der_c(c)),$

where $der_m(m)$ is the damage potential-effort ratio of method m, $der_d(d)$ is the damage potential-effort ratio of the untrusted data d, and $der_c(c)$ is the damage potential-effort ratio of a channel c.

One of the drawbacks of this method is that the source code must be available in order to be utilized. Another drawback with this technique is because the value is 3 dimensional, two systems may be incomparable (i.e., one system will not dominate another system on every dimension, only on one or two dimensions). This issue is captured by Schneider [5] in general comments about measuring security. Schneider states that security is multidimensional and consequently makes comparing systems difficult as one system may be stronger than another system in one dimension and not others. Furthermore, this method does not consider all possible system resources to define the attack surfaces instead it focuses on only the relevant subset of resource types which are more likely to be used in attacks.

AS can be computed via static analysis of a network and is therefore an architectural-based metric. Because a damage potential-effort ratio is being used, effort must be estimated. This property makes AS a complexity-based metrics as well.

3.3.13 Shortest Path (SP) Metric

The Shortest Path metric represents the length of the smallest attack path [9, 52]. The smallest attack path has the shortest distance from an attacker's initial state to the attacker 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 time or resources it may take to exploit vulnerabilities [35, 37]. Resources of an attacker may include, but are not limited to, tenacity, skills, and money [26]. These resources affect an attacker's 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 equation 3.1.

$$SP(G) = \min(l(p_1), l(p_2), ..., l(p_k)) \tag{3.1}$$

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

There are drawbacks to this security metric. In [52], Ortalo et al., denotes 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 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 Figure 3.1 and Figure 3.2 correspond to the attack graphs generated for S_i and S_j respectively. 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 7 different way 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 vulnerabilities from condition s to condition g. Any of the paths of G_i produces a shortest path value of 1 exploit. G_j shows that the path s, $v1$, g produces its shortest path value. Thus, $SP(G_i) = SP(G_j) = 1$ exploit. However, structurally, these attack graphs are relevantly distinct, and we maintain that the security of the two represented systems are also relevantly distinct.

With the exception of its shortest path, G_j has paths that are strictly longer than those in G_i. Recursive application of the Shortest Path metric on subgraphs (i.e., attack paths ordered by path lengths) 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. Such degradation corresponds to the path of least resistance becoming even less resistant. 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

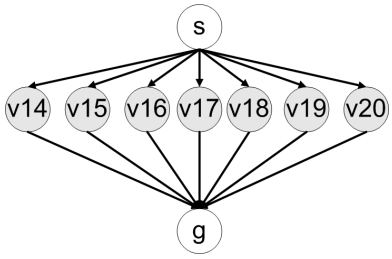

Fig. 3.1.: Attack Graph G_i with vulnerabilities 14 through 20

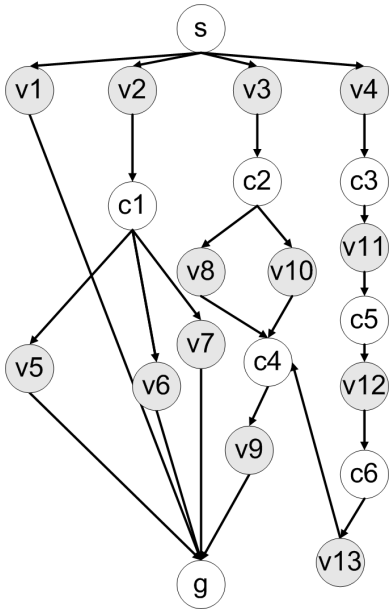

Fig. 3.2.: Attack Graph G_j with vulnerabilities 1 through 13

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 to effectively increase the effort an attacker must expend to reach the 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. 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.

Because SP is to deployed in an attack graph, SP is an attack graph-based security metric. SP is also a complexity-based metric because it attempts to capture attack difficulty.

3.3.14 Number of Paths (NP) Metric

The Number of Paths metric is a value that represents the number of ways an attacker can violate a network's security policy [52]. This security metric expresses the number of attack paths that exist within a given attack graph. The Number of Paths metric is designed to express how exposed a network is to attack. A larger Number of Paths metric suggests a more exposed network. 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 equation 3.2.

$$NP(G) = |p_1, p_2, ..., p_k| \tag{3.2}$$

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.

A drawback of this approach is that attacking 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 one attack graph has 20 attack paths and another attack graph has 1 attack path, the latter attack graph may not necessarily be more secure than the former attack graph. While the former attack graph 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 the latter attack graph. However, there is no known way for making such quantitative assertions regarding effort in practice.

In [52], Ortalo et al. notes that the Number of Paths metric is overly sensitive and unreliable. Although, the sensitivity of the Number of Paths metric is negatively regarded in [52], 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. We

assert that the Number of Paths metric can be made more reliable (see Section 4.4). 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 [12,15,53,54]. 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 domain, then the effect of the countermeasures will leave the attack graph unchanged.

Because NP is deployed on an attack graph, NP is an attack graph-based security metric.

3.3.15 Mean of Path Lengths (MPL) Metric

In [53], 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 must 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. 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 monitoring

security controls. 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.

$$MPL(G) = \frac{\Sigma_i l(p_i)}{NP(G)} \tag{3.3}$$

Despite the benefits this metric provides, there are some shortcomings in its use. If vulnerabilities 15 through 20 are removed from the attack graph in Figure 3.1, 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. And, given our definition of attack path length, 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.

Because MPL is deployed on an attack graph, MPL is an attack graph-based security metric. Also, since MPL attempts to capture typical effort or difficulty required by an attacker, this metric is also a complexity-based metric.

3.3.16 Network Compromise Percentage (NCP) Metric

The Network Compromise Percentage (NCP) metric is a security metric that Lippmann et al. proposed in [55]. 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 equation for the NCP metric is given below.

$$NCP(G) = 100 \times \frac{\sum_{c \in C \subseteq H} c.v}{\sum_{h \in H} h.v} \tag{3.4}$$

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 c. NCP integrates coarse changes in network security. That is, if there is an increase or decrease in the number of hosts that are deemed compromised, the NCP metric will reflect this change in security. However, if vulnerabilities increase at the host level, the NCP metric will be unable to detect this change. That is to say, whether an attacker can exploit a host with a single vulnerability or 10 different vulnerabilities, the NCP metric will represent these two scenarios as the same provided that the attacker can reach the exact same set of hosts with no additional new hosts added to the set. The host with more vulnerabilities in the above example should constitute to a less secure network assuming all other elements of the network remain unchanged. This example indicates how this metric may lead to ineffective usage of resources as countermeasure effects may not be captured by this metric.

Because NCP is deployed on an attack graph, NCP is an attack graph-based security metric.

3.3.17 Weakest Adversary (WA) Metric

Pamula et al. propose the Weakest Adversary metric in [56]. 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 some permutation of the weakest part of the network. The intuition of the metric is that one's network is no stronger than the weakest adversary, (i.e., 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. The weakest adversary metric is given in the equation below.

$$WA(G) = \{W | W \subseteq A \wedge \gamma(W) \leq \gamma(W')\} \tag{3.5}$$

A represents the set of initial conditions that give rise to the successful attack paths. A corresponds to *potential* nodes that could exist at depth 0 in G. The set W corresponds to the *actual* nodes at depth 0 of G. γ is the function the security engineer would have to implement in order to have a $\leq$ relation between subsets of A. W' corresponds to every other possible set of actual initial conditions that is not the same as W.

An issue with this approach, is that this metric may not pick up on changes that happen to internal nodes in the attack graph. Initial attacker conditions give little insight about vulnerabilities that must be exploited within a network to reach a target. Moreover, in our analysis, we assume that attacker initial conditions are known and set by the security engineer.

Because WA is deployed on an attack graph, WA is an attack graph-based security metric.

3.3.18 Attack Resistance (AR) Metric

In [25], Wang et al. propose the use of the Attack Resistance metric. The authors state that a metric should have certain qualities. One is that the security metric for the network should never be larger than its attack path of least effort. Another quality is that a security metric should recognize that different attacks require different levels of time and effort (difficulty). Another quality is that the security of an attack graph with multiple attack paths is less secure than one with only one of the paths. The last observation made is that the realization of one exploit could affect another exploit's resistance value. The authors provide a general framework that has disjunctive and conjunctive operators. The operators are to be defined so as to adhere to the four qualities mentioned above. An example with real numbers defined the disjunction operator as the reciprocal of the sum of the reciprocal of individual resistance values. The conjunctive operator was defined as the sum of individual resistance values. Below is the equation for the attack resistance metric using real numbers. Let nodes e_j, e_k, and e_l (when present) be nodes that are incident on e_i.

$$R(e_i) = \begin{cases} r(e_i) & \text{if no edges are incident on } e_i; \\ r(e_i) + R(e_j) & \text{if nodes } e_i \text{and } e_j \text{ are conjunctive;} \\ r(e_i) + 1/(R(e_k)^{-1} + R(e_l)^{-1}) & \text{if nodes } e_k \text{and } e_l \text{ are disjunctive.} \end{cases}$$

The function r represents the difficulty associated with an exploit e_m. The higher the value of r, the more difficult the exploit is for an attacker to exploit. R represent the cumulative resistance of an exploit e_m by taking into account all resistance values for ancestors of e_m.

The following example illustrates why it is difficult to assess the values produced by the attack resistance metric. Let Figure 3.3 and Figure 3.4 be attack graphs for two distinct network configurations for the same network. The attack resistance of each exploit e_1, e_2, e_3, e_4, and e_5 is 1. The attack graph in Figure 3.3 and the attack graph in Figure 3.4 have an attack resistance of 2. According to the attack resistance metric, these two network configurations have the same ability to resist

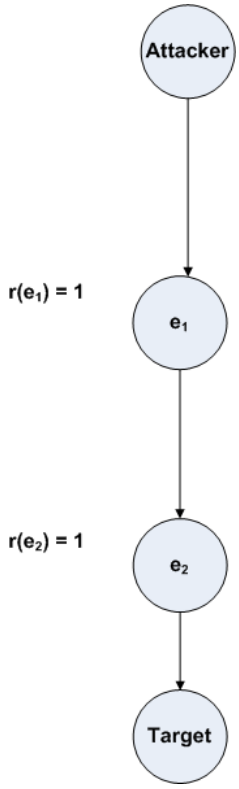

Fig. 3.3.: A single-path exploit dependency attack graph

attack and more specifically have the same level of security. However, this conclusion is erroneous. The attack graph in Figure 3.3 only requires two exploits to be realized in order for the attacker to reach his goal state. However, in the attack graph in Figure 3.4, the attacker in either attack path would have to realize three exploits in order to reach his target state.

In [57] Wang et al. extends the framework with parameters for node significance and reconfiguration costs. No guidance is given as to how resistance values should be assigned. Because the security metric is reduced to a single number, making decisions or taking actions based on a provided attack resistance value may be difficult.

Because AR is deployed on an attack graph, AR is an attack graph-based security metric. Also, because this metric requires the assignment of complexity values to vulnerabilities, this metric is also a complexity-based metric.

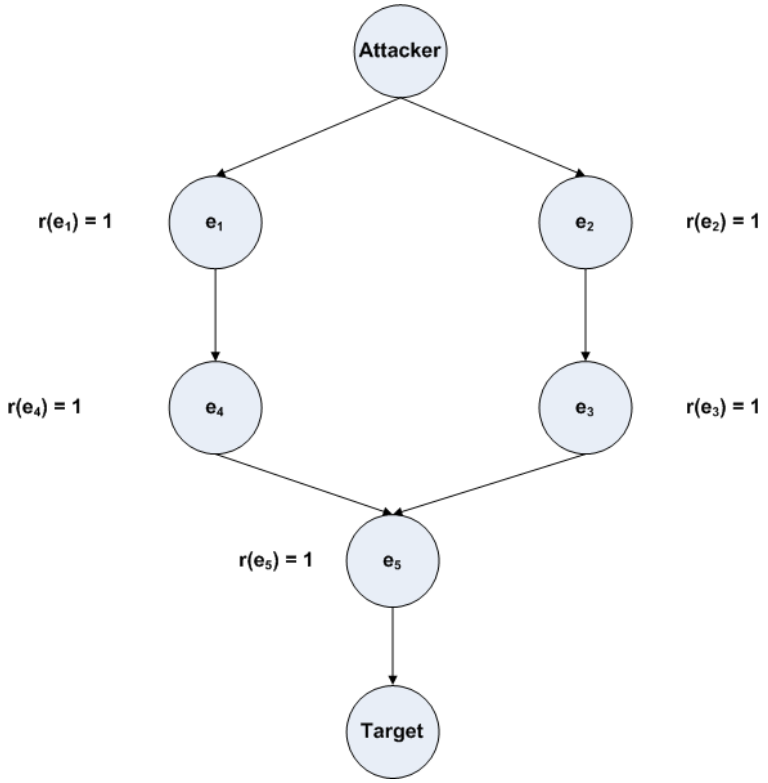

Fig. 3.4.: A two-path exploit dependency attack graph

3.3.19 Probabilistic Reliability Analysis (PRA) Metric

In [10], Sheyner et al. propose the Probabilistic Reliability Analysis metric. Assuming that a security engineer knows the transition probabilities of the attack graph, a Markov Decision Process (MDP) value iteration algorithm can be used to identify the optimal path for the attacker (or the path of least resistance for the attacker). In this approach, goal states are assigned a benefit value of 1 and every other node is a assigned a value of 0. The MDP algorithm is then applied and yields the most probable path the attacker will take: the attack path of least resitance. In [58], Jha proposed the use of probabilistic attack graphs that have probabilistic states and nondeterministic states. Hence, decisions are made at nondeterministic nodes by choosing the transition that will result in the optimal path for the attacker. If $\sum_{s \in S_0} V^*(s)$ represents the attacker's optimal path, then worst case reliability of the network is

1 - $\sum_{s \in S_0} V^*(s)$. This worst case reliability serves as the security metric. There is also a notion of a cumulative score for exploits and conditions. The cumulative score represents that the likelihood that the attacker can reach an exploit or conditions from an entry point. The equation is given below.

$P(e) = p(e) \times \Pi_{c \in Pre(e)} P(c)$

$p(e)$ refers to the probability of exploit e being exploited. $P(c)$ refers to the cumulative probability of condition c being exploited. $Pre(e)$ refers to the preconditions of exploit e.

Because PRA is deployed on an attack graph, PRA is an attack graph-based security metric. Because this metric requires the assignment of probabilities, this metric is also a probability-based metric.

Bayesian Network (BN) Metric

In [26], Dantu and Kolan propose using Bayesian networks to calculate the vulnerability level of important assets. Given an attack graph, the authors posit that attackers of different profiles will possess different characteristics (e.g., skill, tenacity, cost) that will enable them to only be able to attack different portions of the network. Hence, each profile will have an attack graph that looks different. Given the conditional probabilities of the nodes in the subgraphs, some evidence, and the Bayesian theorem, probabilities for the attack subgraph can be updated.

Because BN is deployed on an attack graph, BN is an attack graph-based security metric. Also, because BN requires the assignment of likelihoods, this metric is a probability-based metric.

3.3.20 Attack Graph-based Probabilistic (AGP) Metric

In [59], Wang et al. propose an the Attack Graph-based Probabilistic security metric. Each exploit or condition in the attack graph is assigned a score. The score associated with exploits represents the likelihood of an attacker exploiting the exploit

given that all prerequisite conditions are satisfied. This expression is represented by $p(e)$, where e is the exploit. Scores for conditions are assigned a value 1, to show that they are always satisfied. This expression is represented by $p(c)$, where c is the condition. There is also a cumulative score for exploits and conditions. Cumulative scores corresponds to the likelihood of an attacker reaching an exploit or condition from an entry point. The equation for this is given below.

$$P(e) = p(e) \times \Pi_{c \in Pre(e)} P(c)$$

$$P(c) = \begin{cases} p(c) & \text{if } c \in Init \\ p(c) \times \oplus_{e \in R(c) P(e)} & \text{otherwise} \end{cases}$$

$Pre(e)$ corresponds to the preconditions for an exploit e. $Init$ corresponds to the set of initial conditions in the attack graph. $\oplus$ is notation to capture the recursive nature of the calculation. $\oplus P(e) = P(e)$. For two sets E_1 and E_2 that are subsets of the total set of exploits E, we have $\oplus(E_1 \cup E_2) = \oplus E_1 + \oplus E_2 - \oplus E_1 \times \oplus E_2$.

Because AGP is deployed on an attack graph, AGP is an attack graph-based security metric. Because AGP uses probabilities to obtain its value, it is also a probability-based metric.

3.3.21 Exploitability (EX) Metric

Balzarotti et al. [60] propose the Exploitability metric. After an attack tree is generated, an exploit dependency graph is created. Exploitability values between 0 and 10 are assigned to the different exploits. A value of 0 means that the vulnerability cannot be exploited. Note that the value range is artificially restricted. An exploitability value is assigned to each vulnerability independent of any other vulnerability. This notion is represented by the following expression $E(v_1)$. The exploitability values are assigned to exploits that are conditioned on some other exploit. This notion is represented by the following expression: $E(v_1|v_2)$. Once these values have been assigned, the following formula is to be applied iteratively:

$$\forall v_1 \in V, (v_1, v_2) \in D : E(v) = max(E_0(v_1), min(E(v_1), E(v_1|v_2)))$$

Table 3.1: Classification of network security metrics proposed in the literature

Classification	Metric
Architectural-based	SVI, ACC, LW, TVM, PVM, AP, AS MTTF, MTFF
Attack Graph-based	SP, NP MPL, NCP WA, AR, PRA BN, AGP, EX
Performance-based	FPR, FNR, WF, MTTR MTBF, MDT, MTBMA, MMT
Time-based	MTTB, WF, MTTR, MTTF, MTFF, MTBF, MDT, MTBMA, MMT
Probability/Complexity-based	SVI, MTTF, MTFF PVM, AP, AR, PRA, BN, AGP, EX, AS

D represents the edges in the exploit-dependency graph. This formula says that the attacker will choose the vulnerability that is easiest to exploit. However, $min(E(v_1), E(v_1|v_2))$ constrain attack paths that use dependencies by the vulnerability that is harder to exploit. This metric differs from other metrics that we have examined in this section because it does not produce a value for the network that is being evaluated. Thus, in later analyses of attack graph-based security metrics, we exclude this metric.

Because EX is deployed on an attack graph, EX is an attack graph-based security metric. This metric is also a complexity-based metric because vulnerabilities must be assigned difficulty values.

4. ATTACK GRAPH-BASED SECURITY METRIC AGGREGATION

In this chapter, we outline our method for aggregating attack graph-based security metrics to evaluate networks. However, before specifying our algorithm for combining multiple security metrics, we propose approaches for handling attack path complexity and propose a suite of attack graph-based security metrics. Because some attack graph-based security metrics use the complexity associated with an attack path in their measurements, we must provide a consistent model of complexity. We use *difficulty* and *complexity* synonymously in this dissertation. We present the model of difficulty used in this dissertation in the next section. In section 4.2, we detail five attack graph-based security metrics that compliment previously proposed attack graph-based security metrics. We detail how these metrics can be used together in section 4.4 where we specify our algorithm for attack graph-based security metric combination.

4.1 Addressing Attack Path Complexity

Attack graph-based security metrics that require the assignment of complexity values to vulnerabilities, in practice, default to assigning qualitative values to vulnerabilities. When such values are used, we lose the ability to leverage mathematical algebra to manipulate subsequent vulnerability complexity values. In this section, we propose two approaches to obtaining complexity values for an attack path. The first approach is the Limiting Factor method. The Limiting Factor method holds that what determines whether an attacker reaches a desired goal state is the most difficult vulnerability to exploit along that path. The second method for addressing complexity is the Kolmogorov Complexity approach. The Kolmogorov Complexity approach

provides a flexible framework that allows for creating qualitative or quantitative vulnerability complexity values. If a qualitative interpretation is taken, then the metrics used should not use any algebra or arithmetic operators. Because our (and others') attack graph-based security metrics use algebraic and arithmetic operators, we use a quantitative Kolmogorov complexity approach in this dissertation.

4.1.1 Limiting Factor Approach

The Limiting Factor approach posits that the complexity of an attack path is the difficulty associated with the vulnerability that is most difficult to exploit. This method makes the assumption that if an attacker can exploit the most difficult vulnerability on an attack path, the attacker should be able to the exploit the other less complicated vulnerabilities on the attack path. This approach makes a worst case assumption that the attacker can exploit all vulnerabilities in a network. However, the effort the attacker will have to exert or the skill level required will commensurate with the most difficult vulnerability the attacker must exploit along the chosen attack path.

The Limiting Factor approach decreases the amount of effort a security engineer would have to exert in assigning complexity values. This approach also decreases the computation load of computing complexity values at each node along a path. Using the Limiting Factor approach, any attack graph-based security metric that uses complexity would need to examine, at most one value per attack path.

The limiting factor method is given by the following:

$$\text{Diff}(p) = \max(\cup \text{Diff}(v^p)) \tag{4.1}$$

$\text{Diff}(v^p)$ produces the difficulty value assigned to exploiting vulnerability v that belongs to attack path p. The maximum difficulty value of the union of all such vulnerability assignments is assigned to $\text{Diff}(p)$. $\text{Diff}(p)$ corresponds to the *length* function that attack graph-based security metrics using complexity-based approach

use to measure attack paths (e.g., Shortest Path metric, Mean of Path Lengths metric). The drawback of this approach is that it assumes that skill level is linear, where higher skill level subsumes lower skill level. This assumption implies that an attacker with high skill level can exploit *every* vulnerability deemed requiring medium or low skill level. However, an attacker may have high knowledge of one area (e.g., network protocols) and poor knowledge of another area (e.g., Windows operating systems). This difference in knowledge could cause an attacker to fail to exploit a vulnerability that is considered to require low or medium skill level.

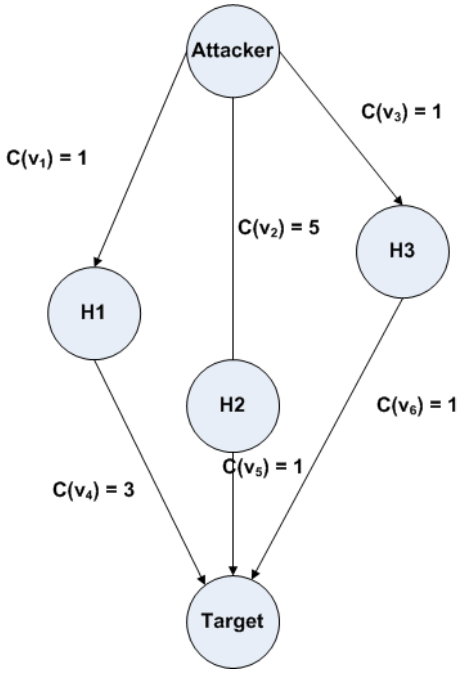

Fig. 4.1.: An attack graph with complexity values C(v_i) assigned to vulnerabilities

If we apply the Limiting Factor to the attack graph in Figure 4.1, the Limiting Factor produces complexity values that correspond to the most complex vulnerability on each path. C(v_i) represents the complexity value the security engineer assigned vulnerability v_i. Higher values correspond to more difficult vulnerabilities that are more difficulty to exploit. Because the Limiting Factor approach takes the largest complexity value on each path, the Limiting Factor approach would produce the

following complexity values for the attack paths in Figure 4.1: 3, 5, and 1. The values would then be treated as the path lengths of the attack graph.

4.1.2 A Kolmogorov Complexity Approach

Kolmogorov Complexity determines the string complexity using the size of the smallest program that can produce the string [61]. Let K represent the Kolmogorov Complexity function. In comparing two strings, x_1 and x_2, if $K(x_1) < K(x_2)$, then x_1 is less complex than x_2 because a smaller program can be used to describe x_1. The concept of using Kolmogorov Complexity to monitor security was first promulgated by Evans et al. in [62]. We use Kolmogorov Complexity to provide a systematic approach to determine complexity of attack paths in an attack graph. Although it is impossible to determine a lower bound K [61], a formal approach to determine complexity of attack paths is useful as it provides consistency in comparing measurements. Below we provide a language inspired by the language of regular expressions.

Alphabet

A corresponds to the exploits found in all attack graphs being considered.

Constants

ϵ corresponds to the empty string.

$e_i \in A$ denotes an exploit from one of the attack graphs being considered.

$\emptyset$ corresponds to the empty set.

Operations

Let S and T be two strings comprised of characters from A. Also, let E_1 and E_2 be expressions of the language.

ST evaluates to the concatenation of string S and T.

() provides priority ordering of evaluation.

$(S)^+$ the expression S *may* repeat more than one time but must appear once

S^k repeat S k times.

$E_1^{[k]}E_2$ evaluates to inserting E_1 at index k in E_2, where the first character in E_2 corresponds to the zeroth index. This can be generalized to $E_1^{[k_1],[k_2],[k_3],\ldots,[k_{n-1}],[k_n]}E_2$. This rule would not be used independently. It would only be used as part of the following two rules.

$E_1^{l,[k]}E_2$ evaluates to concatenating E_1^l to E_2, and inserting E_1 into index k of E_2.

$E_1^{l[k]}E_2$ evaluates to inserting E_1^l into index k of E_2.

This language allows for expressive ways for representing the complexity of attack paths. The attack path in Figure 4.2 can be represented a number of ways. Hi, Attacker, and Target in Figure 4.2 correspond to hosts, and v_j corresponds to vulnerability j. Based on the last two operations (i.e., $E_1^{l,[k]}E_2$ and $E_1^{l[k]}E_2$) a qualitative representation of this path can be represented as $v_1^{3,2[2]}v_2v_3$. An equivalent representation would be $v_1^{3,[2],[2]}v_2v_3$. In both representations, E_1 corresponds to v_1 and E_2 corresponds to v_2v_3. The path length of either representation is 3 nodes. This representation suggests that once the attacker exploits H1, attacking hosts H2, H3, H6, and Target is trivial because they require the same credentials to have their vulnerabilities exploited (e.g., these hosts use the same username and password). Such a representation may also be used if known scripts can be used to exploit vulnerabilities in the system. Ultimately the qualitative representation used depends on the decision of the security engineer. For instance, the attack path can alternatively be represented as $v_1^{3,[2]}v_2v_3v_1$. The path length of this representation is 4 nodes. This representation suggests that the attack path in Figure 4.2 is more secure than the representation initially described. The semantics of this latter representation suggest that while the same information can be used in exploiting hosts H2, H3, H4, and H6, different information is required to exploit Target. If the attack path is represented as

$v_1v_1v_1v_2v_3v_1v_1$, then this would suggest that different credentials are required for compromising each host. This representation corresponds to the quantitative approach used in this dissertation.

In this dissertation, we use only the concatenation operator to construct attack paths. This representation is equivalent to counting the number of exploits that exists along an attack path.

Our Kolmogorov Complexity approach to representing attack path complexity has the ability to represent cycles in the the attack graph. Figure 4.3 shows a infinite number of attack paths because there is a cycle in the nodes displayed. Typically, cycles in the attack graph are ignored. Our approach is robust enough to capture cycles in the graph. This infinite number of attack paths can be captured with $v_1^2(v_1v_2v_3)^+v_1^2$. Instead of ignoring cycles, they can be captured by this representation.

4.2 A Complimentary Suite of Attack Graph-based Security Metrics

While the attack graph-based security metrics mentioned thus far can be useful if used appropriately, our analysis suggests they should be used together. For instance, 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 (along with our metrics), they can give a more comprehensive measure of security. We detail how in section 4.4. In this section, we propose our metrics that are designed to assist a network administrator in determining more relevant properties of the network to determine its security. We propose the following metrics: the Normalized Mean of Path Lengths metric, the Standard Deviation of Path Length metric, the Mode of Path Lengths metric, the Median of Path Lengths metric, and the K-step Capability Accumulation metric.

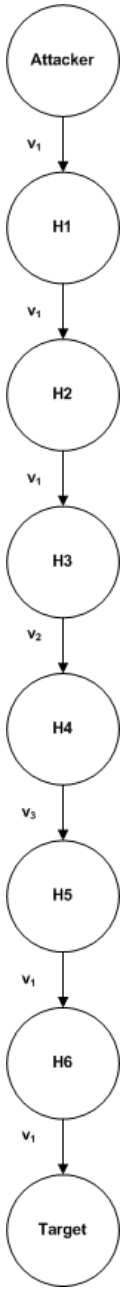

Fig. 4.2.: A single attack path with Attacker, Hi, and Target corresponding to hosts and v_j corresponding to vulnerabilities

4.2.1 Normalized Mean of Path Lengths (NMPL) Metric

The NMPL metric is the Mean of Path Lengths (MPL) metric divided by the Number of Paths (NP) metric. This normalization is critical when comparing two

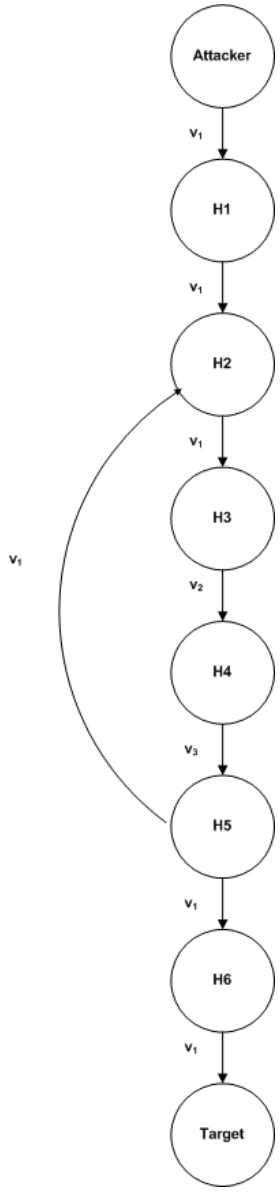

Fig. 4.3.: An attack path containing a cycle

attack graphs to determine which underlying network is more or less secure. Given the attack graphs shown in Figure 4.4 and Figure 4.5, the Mean of Path Lengths metric would conclude that both underlying networks have the same level of security. Both attack graphs have a mean path length of 1 edge. However, by the NMPL metric the attack graph in Figure 4.5 is less secure than the attack graph in Figure 4.4. The NMPL for the attack graph in Figure 4.5 has a NMPL of 0.20 edges, while

the NMPL for the attack graph in Figure 4.4 has a NMPL of 1 edge. Further support for usage of the NMPL is provided in section 5.1.3. The equation is given below.

$$NMPL(G) = \frac{MPL(G)}{NP(G)} \tag{4.2}$$

Fig. 4.4.: A single path attack graph where the attacker has direct access to the target

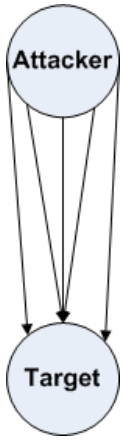

Fig. 4.5.: An attack graph where the attacker has 5 ways of directly access the target

4.2.2 Standard Deviation of Path Lengths (SDPL) Metric

The Standard Deviation of Path Lengths metric, when added and subtracted from the Mean of Path Lengths metric, gives the range containing typical attack path lengths. 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 a path length is two standard deviations below the Mean of Path Lengths metric, this path may deserve the attention of the security engineer. However, if 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.

$$SDPL(G) = \sqrt{\frac{\Sigma_i(l(p_i) - MPL(G))^2}{NP(G)}} \tag{4.3}$$

l is the length function that returns the length of an attack path p. The length function we use is simply the number of vulnerabilities encountered along an attack path. The standard deviation for Figure 4.6 and Figure 4.7 are 1.29 edges and 1.41 edges respectively. The attack graph in Figure 4.7 varies more widely about its mean than Figure 4.6. This variation suggests regardless of what the mean attack path length is, there are paths relevantly lower than mean path length. Nonetheless, in determining the security of the network more analysis is required to know where most of this variability exists (see section 4.4).

4.2.3 Mode of Path Lengths (MoPL) 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 [63] to assign an equal probability to each attack path. The Mode of Path Lengths metric suggests a likely amount of effort an attacker

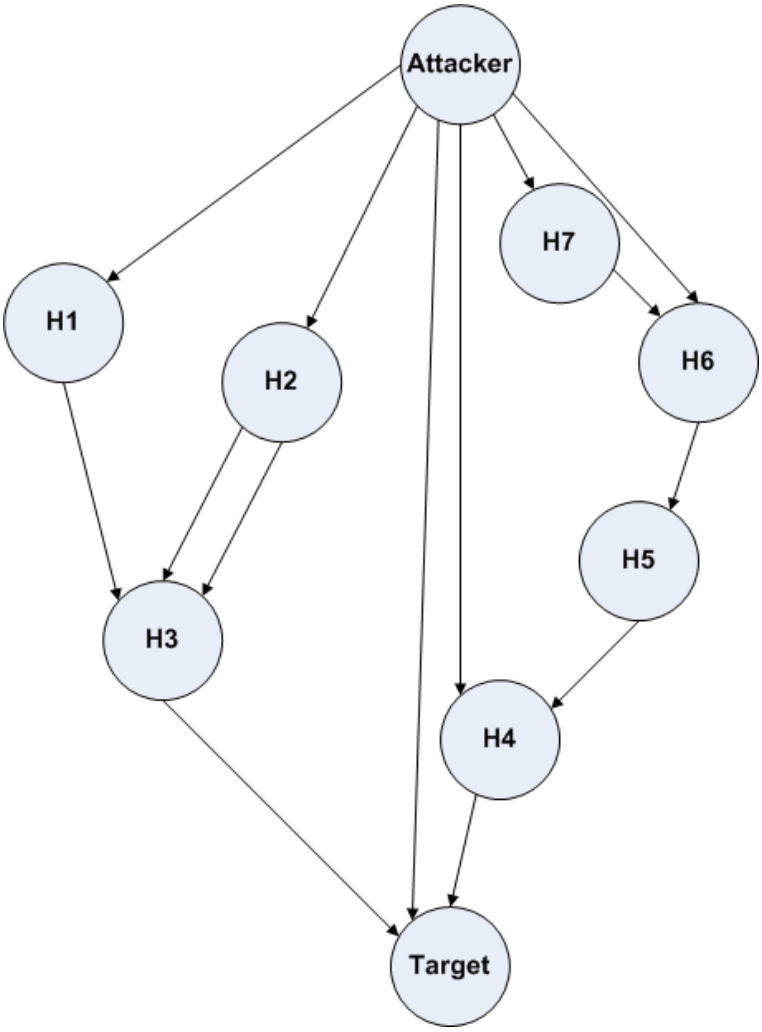

Fig. 4.6.: A condition oriented attack graph with intermediate hosts H1 through H7

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 affected 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.

$$MoPL(G) = f(l(p_1), l(p_2), ..., l(p_k)) \tag{4.4}$$

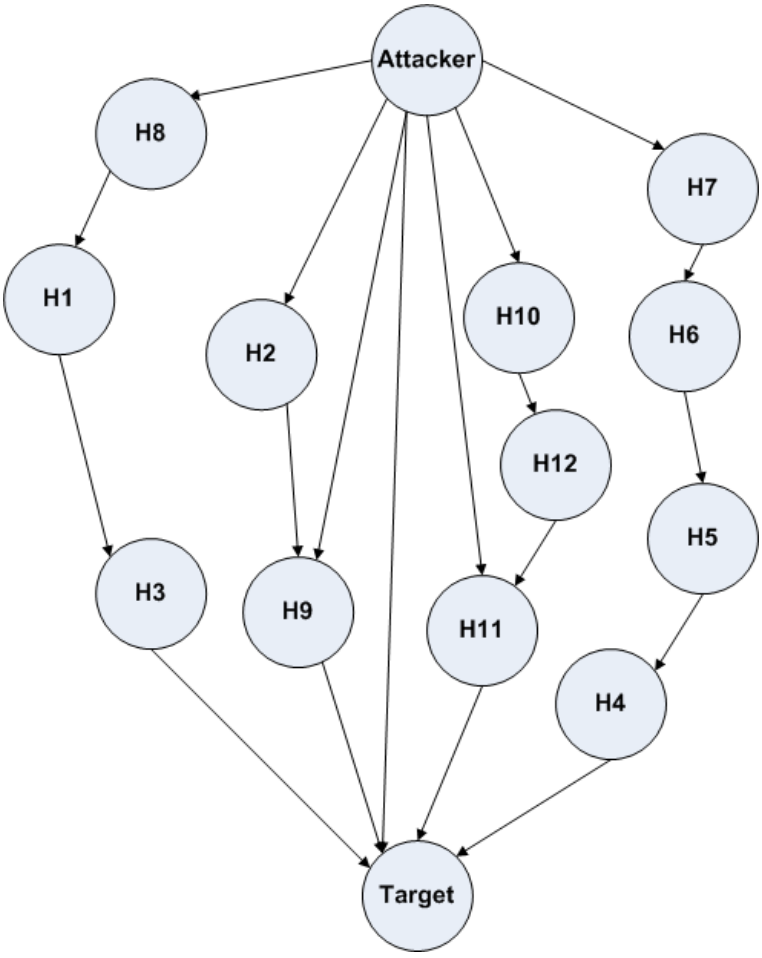

Fig. 4.7.: A condition oriented attack graph with intermediate hosts H1 through H12

The mode for Figure 4.6 and Figure 4.7 are (3) and (2, 4) respectively. In attempting to understand the security of the system with this metric, a security engineer has many options. However, any option would require that more analysis be done. That is, inherently, the mode does not reveal much about the security of the network. However, if the frequency of the of the modes in both attack graphs are known, then better assessments can be made (see section 4.4).

4.2.4 Median of Path Lengths (MePL) 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 affect 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. Note that $l(p_i)_q \leq l(p_j)_{q+1}$, which 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 $q+1$.

$$MePL(G) = \begin{cases} l(p_i)_{\lceil \frac{k}{2} \rceil} & : \text{k is odd} \\ \frac{1}{2}(l(p_i)_{\frac{k}{2}} + l(p_j)_{\frac{k}{2}+1}) & : \text{k is even} \end{cases} \tag{4.5}$$

k in the above equation is the number of attack paths in the attack graph G. In determining the security of the system, the median provides a point to perform further analysis. In Figure 4.6 and Figure 4.7 the median is 3. In general, the Median of Path Lengths metric will correspond to the path length that represents the 50th percentile of all path length values. Hence, an analyst should have keen interest in the values that fall below the median. More specifically, if comparing two attack graphs, the one that has more longer paths below the median may be deemed more secure.

4.2.5 K-step Capability Accumulation (KCA) Metric

The K-step Condition Accumulation (KCA) metric specifies the "power" the attacker can obtain on a network in K steps. Practically, "power" represents the capability an attacker attains. Capabilities are controlled by access controls. Therefore,

power may be represented by the privilege(s) the attacker attains on a machine. Therefore, if an attacker can obtain more capabilities on Sys_1 in K steps, than the attacker can on Sys_2 in K steps, then Sys_2 is more secure than Sys_1. More generally, if an attacker can obtain strictly more power in Sys_1 than in Sys_2 in K or less steps, then Sys_2 is more secure than Sys_1. The formalization of this metric is given below.

$$Cap_h(G) = \cup_h capabilities(n) \tag{4.6}$$

$$KCA_k(G) = \cup_{i=0}^{k} Cap_i(G) \tag{4.7}$$

capabilities(n) returns the set of capabilities available at node n. If more granularity is desired, the *capabilities* function may be extended with a set of vulnerabilities as an input parameter. Different vulnerabilities may provide different capabilities for an attacker. Thus to provide this granularity, capabilities can be extended as *capabilities*$(\{v_1, v_2, v_3..., v_{m-1}, v_m\}, n)$. Vulnerabilities v_1 through v_m correspond to the vulnerabilities that an attacker may exploit to violate a security policy at node n.

$Cap_h(G)$ represents the capabilities obtained at level h in attack graph G. h represents the distance from the attacker's initial state. If we were to apply the KCA metric to the attack graph in Figure 4.8 we would obtain different values at each level of the attack graph. The nodes of the attack graph correspond to hosts and access levels. The labels have the format of *host:access level*. U corresponds to user level access. A refers to administrator-level access. Thus, Cap_1 = user level access on H1 and H2, administrator level access on H3. $Cap_2 = Cap_1$ and administrator level access on H1 and H2, and administrator level access on H4. $Cap_3 = Cap_2$ and administrator level access on the Target host. The attacker can accumulate all privileges in the network in 3 levels. If we wanted to examine two network configurations using KCA, then we could examine KCA at each level. Observe the attack graph in Figure 4.9. The KCA at level 1 for the attack graph in Figures 4.8 and 4.9 is the same. However, we can see that in Figure 4.9 that the Target can be reached in two steps. Assuming

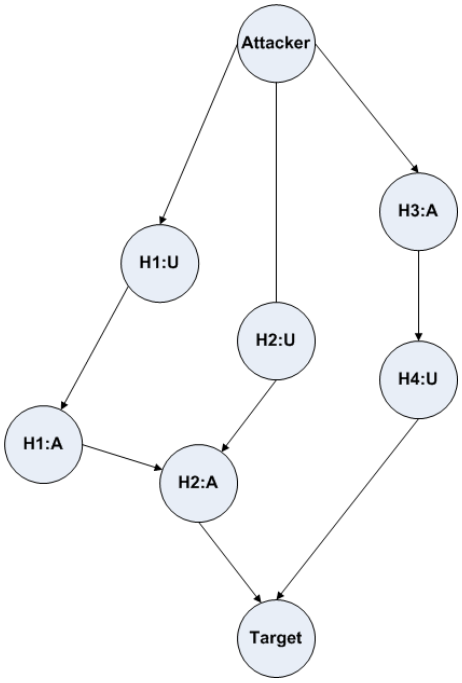

Fig. 4.8.: A 3 path attack graph

that this is the most important host to protect, then the attack graph in Figure 4.9 is less secure because the attacker is able to obtain more power in fewer steps. Although there is a goal state in the attack graphs evaluated in the above example, this metric can be applied to attack graphs with no goal states.

When there is a goal state and the semantics of the attack graph are such that it has all of the weight, or all the valuable power, the K-step Capability Accumulation metric may degenerate to the Shortest Path metric. For instance, observing the attack graphs in Figure 4.6 and 4.7, both have the attacker reaching the goal state in single step. If the non-attacker nodes do not matter with respect to the target node, then using the Shortest Path metric would be sufficient to attain this result. However, if other nodes are perceived as being relevant to the network's security, then the KCA metric can be used to obtain more information about security of the network.

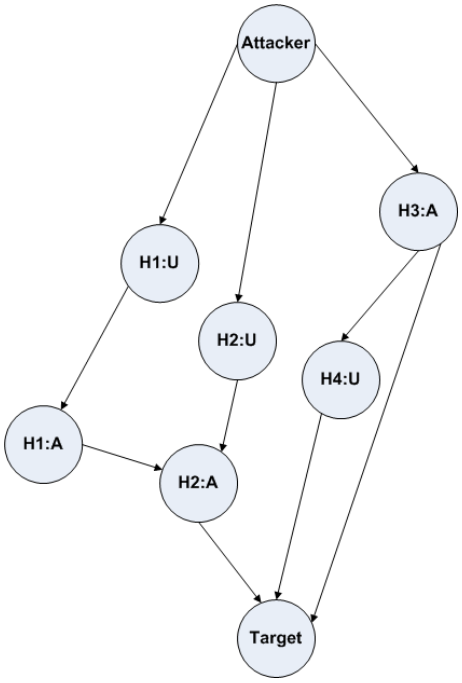

Fig. 4.9.: A 4 path attack graph where nodes correspond states that give the host name and access level and as formatted as host:access level

4.3 Attack Graph-based Security Metrics and Their Applicable Attack Graphs

Table 4.1 shows what attack graph-based security metrics can be applied to the various types of attack graphs. The only types of attack graphs that cannot be used with all attack graph-based security metrics are the Host Compromise Attack Graph, the Predictive Attack Graph, and the Node-predictive Attack Graph. One commonality among each of these attack graphs is that they are all not goal-oriented and cannot be converted into a goal-oriented representation.

4.4 Using Multiple Security Metrics

We propose a methodology for using attack graph-based security metrics together harmoniously. We assert that there is no one security metric that will divulge all

Table 4.1: What security metrics can be applied to which attack graphs

Attack Graph	Applicable Security Metrics
NuSMV	All AG-based Security Metrics
Coordinated	All AG-based Security Metrics
Full	All AG-based Security Metrics
Host-compromised	Network Compromise Percentage
Predictive	Shortest Path Metric, K-Step Capability Accumulation Metric, Network Compromise Percentage Metric
Node-predictive	Shortest Path Metric, Network Compromise Percentage Metric
Multiple Prerequisites	All AG-based Security Metrics (double check)
Logical	All AG-based Security Metrics
Hybrid-oriented	All AG-based Security Metrics

there is to know about a network's security. Given this assertion, an important goal in evaluating network security is having a method for combining the usage of appropriate metrics to reach a decision about the security of a network. In this section, we describe decision metrics and assistive metrics and provide the algorithm for combining these metrics.

4.4.1 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. They should be applied first in case all decision metrics agree regarding the network's security. If they agree there is no need to involve other metrics. The decision metrics discussed in this paper are: the Shortest Path metric, the Number of Paths metric, the Normalized Mean of Path Lengths metric, the Network Compromise Percentage metric, the K-step Condition Accumulation metric, the Weakest Adversary metric. These metrics are also shown in the top row of Table 4.2. We further distinguish these metrics by whether they are path analysis metrics or not. Path analysis metrics factor in attack path complexity into its resulting value. Of the decision metrics in Table 4.2, the only metrics that are not path analysis metrics are the Network Compromise Percentage metric and the Weakest Adversary metric. We explain how these metrics can be used in the next section. For remainder of this section, when referring to decision metrics, we are referring the path analysis-based security 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:

- Which attack graph produces the shortest attack path requiring the most effort?
- Which attack graph gives the attacker the least number of ways of violating a security policy?
- 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 three 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 security engineer could use an evaluation approach where the attack graph satisfying the question deemed most important is considered more secure. Alternatively, the security engineer could assume each question is equally important, 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.

An 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 4.4.2) in case assistive metrics are required (line 17). 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, Algorithm 3). The decide function then determines which attack graph is most secure with respect to metric m (lines 4-9).

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. Otherwise further analysis is obtained from assistive metrics (line 15).

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 5.2). However, in the cases when this conclusion cannot be achieved, assistive metrics are required.

4.4.2 Assistive Metrics

Assistive metrics serve as "drill down" metrics. These metrics discover more security-relevant information from the attack graph. Unlike decision metrics, these metrics are not used independently to make determinations regarding which 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 4.2. We have specified our approach to 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 G with subsets of attack paths 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 1 *compareGraphs* function: Algorithm for Using Multiple Metrics to Evaluate Two Attack Graphs

Require: G_1, G_2 {attack graphs to be compared}, M {the set of security metrics to be used for attack graphs}, t {threshold value for Mode of Path Lengths metric}, 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}, R {result set}, m_1 {metric to apply to the two attack graphs}, m_2 {assistive metric}, 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}, m {security metric to apply to G_1 and G_2}, 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** $G_2.name$

Algorithm 4 *isStrictlyDominated* function

Require: R {result set}, 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}, 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 s function, the $keepPathsEqualTo$ function returns an attack graph with paths that have path lengths equal to the Mode Path Length 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 function $keepPathsBelowOrEqualTo$ 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 the typical amount of attack effort that is greater 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 subgraphs 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 11-12). When the Number of Paths metric is applied to the $s(G_1, SDPL)$ and $s(G_2, SDPL)$, the resulting expression captures the number of typical paths exist in the attack graph for one meaning of typical. $MePL'$, the modified Median of Path Lengths metric, is the minimum median of two attack graph 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.

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 subgraphs 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 15-21). 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 a notion of common attack paths in the attack graph. 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.

To come to a decision about which attack graph is most secure, a total ordering for the values in R_a may be established. 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. The security engineer could then apply the NCP or the WA metrics to the full attack graphs of the two attack graphs being compared in order to reach a decision about which network is most secure. However, 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.

4.5 Assessing Two Attack Graphs

Table 4.3 gives the result of applying our network security evaluation approach to attack graphs G_i and G_j. It is evident from the table that by simply looking

Table 4.2: 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
	KCA K-step Condition Accumulation
Assistive	MPL Mean of Path Lengths
	$SDPL$ Standard Deviation of Path Lengths
	$MoPL$ Mode of Path Lengths
	$MePL$ Median of Path Lengths

Algorithm 6 *enlistAssistiveMetrics* function

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

1: **for all** $r.r_1 \in R_d$ **do**
2: **if** $r.r_1$ equals SP **then**
3: **for all** $m \in \{MoPL, SDPL\}$ **do**
4: **if** m equals $MoPL$ **then**
5: $R_a \leftarrow applyMetric(s(G_1, m, t), s(G_2, m, t), R_a, r.r_1, m, >)$
6: **else**
7: $R_a \leftarrow applyMetric(s(G_1, m), s(G_2, m), R_a, r.r_1, m, >)$
8: **end if**
9: **end for**
10: **else if** $r.r_1$ equals NP **then**
11: **for all** $m \in \{MePL', SDPL\}$ **do**
12: $R_a \leftarrow applyMetric(s(G_1, m), s(G_2, m), R_a, r.r_1, m, >)$
13: **end for**
14: **else if** $r.r_1$ equals $NMPL$ **then**
15: **for all** $m \in \{MePL', MoPL, SDPL\}$ **do**
16: **if** m equals $MoPL$ **then**
17: $R_a \leftarrow applyMetric(s(G_1, m, t), s(G_2, m, t), R_a, r.r_1, m, >)$
18: **else**
19: $R_a \leftarrow applyMetric(s(G_1, m), s(G_2, m), R_a, r.r_1, m, >)$
20: **end if**
21: **end for**
22: **end if**
23: **end for**{check for strict domination, majority domination, and equality in R_a, returning (R_d, R_a) if any one check is true otherwise return (R_d, R_a)}

Algorithm 7 s function

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

if t is passed in as a parameter **then**
  if $isBelowThreshold(t, m(G), G)$ **then**
    return $\emptyset$
  end if
end if
if m equals $MoPL$ **then**
  return $G' \leftarrow keepPathsEqualTo(m(G), G)$
else if m equals $SDPL$ **then**
  return $G' \leftarrow keepPathsInRange(\chi(G) - m(G),\ \chi(G) + m(G),\ G)$
else if m equals $MePL'$ **then**
  return $G' \leftarrow keepPathsBelowOrEqualTo(m(G),\ G)$
end if

Table 4.3: Security metric evaluation of G_i and G_j

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

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 Path Length metric for G_j and inspecting the path lengths above and below it, the security engineer can ascertain that it has at least 3 different values (1, 2, and 5 exploits). This reveals that half of the attack paths are equal to the 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 reveal 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 1 exploit and 3 exploits. These values are obtained by adding or subtracting the Standard Deviation metric from Mean of Path Lengths metric. The result of applying the *compareGraphs* to G_i and G_j is shown in Table 4.4. Hence, according to *compareGraphs*, G_j is more secure than G_i.

Table 4.4: The result of applying decision and assistive metrics to G_i and G_j

Security Metric Value	Result
SP on G_i and G_j	G_i and G_j are equally secure
NP on G_i and G_j	G_i and G_j are equally secure
$NMPL$ on G_i and G_j	G_j is more secure
SP on $s(G_i, MoPL, t)$ and $s(G_j, MoPL, t)$	G_j is more secure
SP on $s(G_i, SDPL)$ and $s(G_j, SDPL)$	G_j is more secure
NP on $s(G_i, MePL')$ and $s(G_j, MePL')$	G_j is more secure
NP on $s(G_i, NMPL)$ and $s(G_j, NMPL)$	G_j is more secure
NP on $s(G_i, MoPL, t)$ and $s(G_j, MoPL, t)$	G_j is more secure
$NMPL$ on $s(G_i, MePL')$ and $s(G_j, MePL')$	G_j is more secure
$NMPL$ on $s(G_i, NMPL)$ and $s(G_j, NMPL)$	G_j is more secure
$NMPL$ on $s(G_i, MoPL, t)$ and $s(G_j, MoPL, t)$	G_j is more secure

5. SIMULATION STUDIES

This chapter describes the simulation studies conducted to assess the behavior of various attack graph-based security metrics. More specifically these studies seeks to provide evidence to support characterizations of the attack graph-based metrics discussed in chapters 3 and 4. Recall that assistive metrics (i.e., the Mean of Path Lengths metric, the Standard Deviation of Path Lengths metric, the Median of Path Length metric, the Mode of Path Lengths metric) are used to help the decision metrics reach a decision about which of two network configurations is most secure. As shown in the previous chapter, by identifying relevant subsets of attack paths, these assistive metrics can help resolve conflicts. Nonetheless, these metrics are not to be used as decision metrics and our simulation studies provide further evidence to support our claim. We also show how Algorithm 1 (from the previous chapter) performs on a series of artificial attack graphs.

5.1 Characterization of Attack Graph-based Security Metrics

In this section we provide simulated data that supports the classification used for metrics previously proposed. We initially describe the network model used and then describe how we set up our experiments. We then begin our assessment of attack graph-based security metrics. Where applicable, we specify associated equations that explain the behavior of the metrics being evaluated.

5.1.1 Network Models

In this section, we describe the network models used in our simulation studies. We chose a set of base network models: the Flat network, External-Internal network, and

the External-Demilitarized Zone (DMZ)-Internal network. More sophisticated networks models can be decomposed into these basic network models. For each network we use in our experiments, host A refers to the attacker's host. Thus the attacker is modeled as having an account on host A with connectivity that is dictated by the network model being used. We assume the attacker has the ability to exploit all discovered vulnerabilities in a network. Hosts B through D are intermediate hosts. Host V is the victim host. Thus, users are modeled as having accounts on hosts B through D and V. These machines are assessed under various vulnerability densities (explained later). The victim host is the attacker's target in each scenario. While these network models use hosts, these could also be viewed as subnetworks.

Flat Network

In the Flat network, each host can connect to any other host. If the attacker and target are in the same network then the attacker has direct access to the target host. Any subnetwork of an organization where such connectivity is allowed can be classified as a Flat network. For instance, there may be wireless access points that allow members of the organization to connect to a particular internal subnetwork. An instance of a Flat network model is given below in Figure 5.1.

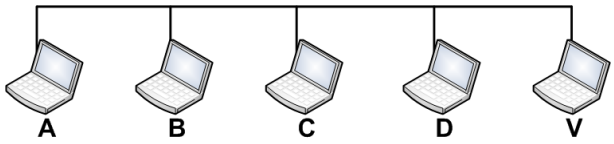

Fig. 5.1.: Flat network model

External-Internal Network

In the External-Internal network, a filtering device disallows connectivity to a subset of addresses and ports in either direction (from the external network to the

internal network or from the internal network to the external network). An organization may put up any number of these filtering devices within a given network. Understanding what happens in the presence of a single filtering device will assist us in understanding larger more complex network models. An example of this model is shown in Figure 5.2. In our experiments, host A can connect directly only to host B. Hosts B, C, D, and V can connect to each other.

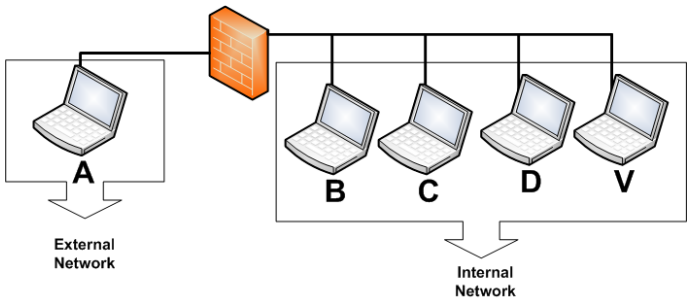

Fig. 5.2.: External-internal network model

External-DMZ-Internal Network

In the External-DMZ-Internal network, there are at two filtering devices: a DMZ filtering device and an internal filtering device. The DMZ filtering device is responsible for filtering connections aimed toward the network. This network model filters connections that come from the external network and are destined for the DMZ or internal networks. The internal filtering device filters connections that come from the DMZ network destined for the internal network. These connections could have ultimately originated from the external network or the DMZ network. An example of the network model is shown in Figure 5.3. This network model can be seen as two External-Internal network models M_1 and M_2. The internal network of M_1 would correspond to the external network of M_2. In other words, the internal network of M_1 would be the same as the external network of M_2. In our experiments, host A can connect directly only to host B. Host B can connect directly to host C. Host C

can connect directly to Hosts B, D, and V. Host D can connect directly to host V. Host V can connect directly to hosts C and D.

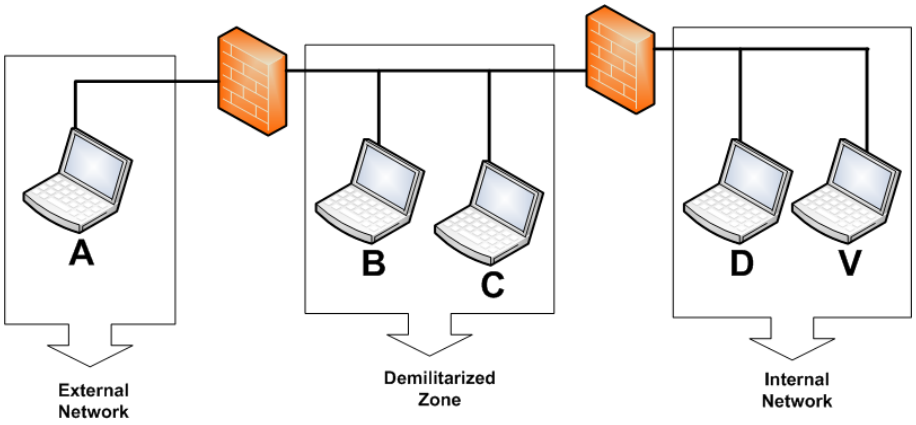

Fig. 5.3.: External-DMZ-internal network model

5.1.2 Experiment Setup

The experiment setup is depicted in Figure 5.4. The Input Generator creates input files for MulVal [64] that creates the corresponding attack graphs. We chose to use MulVal to generate attack graphs because it has the ability to produce attack graphs efficiently and was made accessible by its author [64]. The Input Generator assigns vulnerabilities to hosts in the network. The Attack Graph-based Security Metric Engine uses attack graph-based security metrics to measure attack graphs. For each of attack graph measured, the result of applying each attack graph-based security metric to the attack graph is stored to a database.

We use goal-oriented attack graphs. When machines can connect to each other, they connect on a single port (port 80). When the number of vulnerabilities are modified, they are modified for this port. We assume the vulnerabilities are remotely exploitable. Reachability is the precondition for the vulnerabilities. That is, if an attacker can connect to a machine, the attacker can exploit any vulnerabilities present on the machine. The consequence for the vulnerabilities is that the attacker has the

ability to execute arbitrary code on the compromised machine. We leave the issue of locally exploitable vulnerabilities to future work.

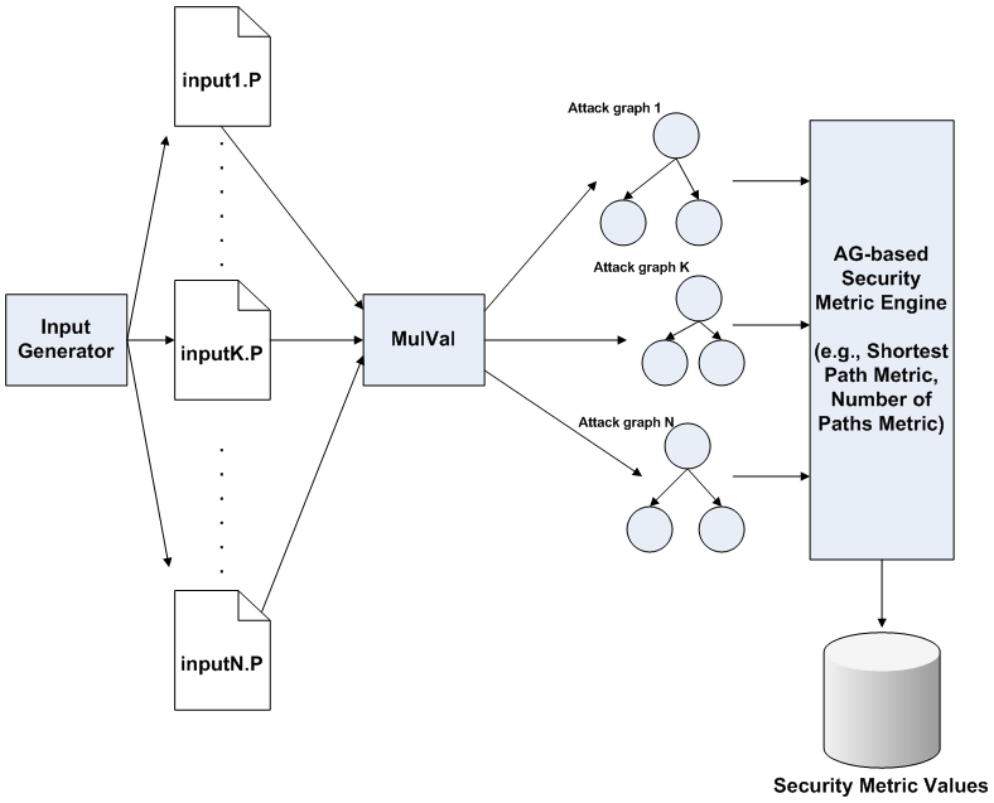

Fig. 5.4.: Experiment framework

5.1.3 Attack Graph-based Security Metric Analysis

In this section, we evaluate attack graph-based security metrics on the aforementioned network models. In the following charts, "Flat" corresponds to the Flat network model; "E-I" corresponds to the External-Internal network model; "E-D-I" corresponds to the External-DMZ-Internal network model. Along the x-axes of the of these charts, there is an assignment of remotely exploitable vulnerabilities to hosts. The assignment is given by the following format: iH, where i is the number of remotely exploitable vulnerabilities assigned to host H.

We evaluate the networks' security incrementally. That is, after host V is evaluated with a vulnerability, we evaluate the security level of the network with hosts V and B having vulnerabilities. Then we evaluate the security level after we add another vulnerable host C. Then we evaluate network security after we add a last vulnerable host D. Once host D is added to the network, we increase the number of vulnerabilities for each host. At each increase, we reevaluate the the attack graph for all attack graph-based security metrics. Figures 5.1, 5.2, and 5.3 correspond to the scenario when all hosts have joined the network in the Flat network, External-Internal network, and External-DMZ-Internal network. We increase the number of remotely exploitable vulnerabilities at host V first. This initial vulnerability assignment is done because there can be no attack graph if host V has no vulnerabilities.

There are two decision path analysis attack graph-based security metrics we do not include in our analysis. The first is the Attack Resistance metric. We do not include the AR metric in this study because the metric is unable to handle all types of attack graphs. Namely, the AR metric cannot deal with scenarios where an attacker may exploit multiple vulnerabilities that exists on a single attack path directly. The second decision path analysis attack graph-based security metric excluded from this analysis is the K-step Condition Accumulation metric. We do not include KCA metric in this study either. This exclusion is due to the granularity of the study. In this study, we are looking at the effects of vulnerabilities at the host level. Thus, the KCA metric would show the hosts compromised as the conditions accumulatd by the attacker. Because of this perspective, the data that would be produced by the KCA metric would be the same as the data produced by the NCP metric. And in this study, we do evaluate the NCP metric.

Shortest Path Metric Assessment

Observing the graph in Figure 5.5, reveals why the Shortest Path metric should not be used for monitoring the security of a network. In each type of network,

as the network increases in the number of remotely exploitable vulnerabilities, the Shortest Path metric remains constant. This data provides further support for our recommendations for appropriate usage of the Shortest Path metric in section 3.3.13.

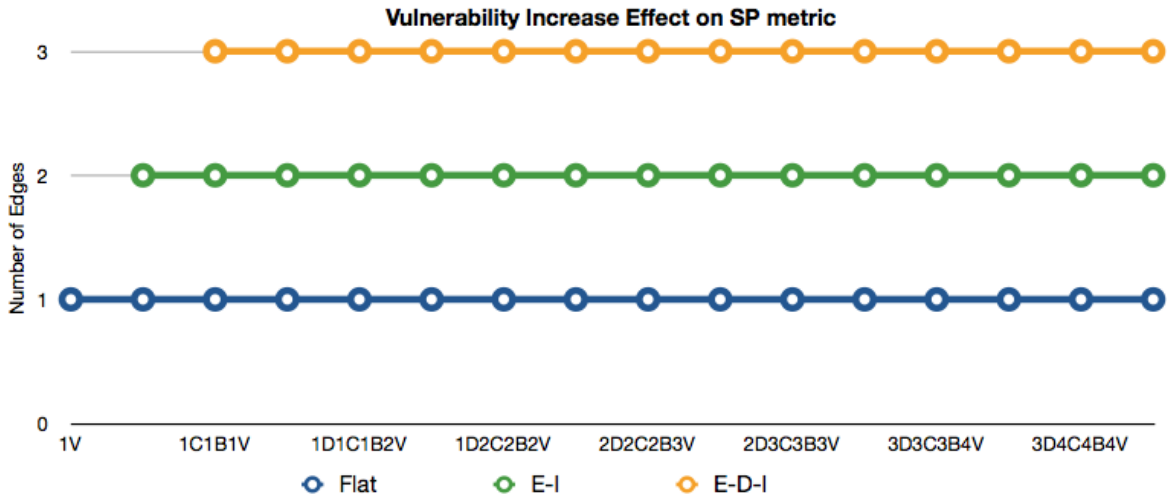

Fig. 5.5.: The effect of vulnerability increase on the shortest path metric under different network models

The usage of a firewall(s) before the target increases the length of the path an attacker must traverse to achieve the goal. Thus, this metric provides quantitative support for the well-known defense in depth network protection strategy. Moreover, by looking at the values for External-Internal and the External-DMZ-Internal network models, the values are not defined until the attacker can exploit enough vulnerabilities to reach the target. This observation suggests another strategy for protecting vulnerable hosts. The strategy is if a host is known to be vulnerable or more vulnerable than other hosts in the network and susceptible to attack, then access to the machine should be reduced. This countermeasure can be accomplished by shutting the service down on the port or instrumenting the service to increase the effort required to access the service.

Number of Paths Metric Assessment

Because hosts existing in the same protection domain can reach each other, cycles emerge in the attack graph. However, if there is a single cycle in an attack graph, then the Number of Paths metric would produce a value of infinity, as there would be infinite paths in the attack graph. To deal with this issue, the Number of Paths metric breaks cycles in the attack graph when performing measurements.

In our Number of Paths metric measurement, we compute the metric by determining the number of paths at each node in a breadth-first-search manner. We start the measurement from the attacker's target state progressing toward the attacker's initial state. Edges that point to previously discovered nodes are ignored.

Irrespective of the network model, the trend of the Number of Paths metric is to increase exponentially as the number of vulnerabilities in the network increases linearly in Figure 5.6. This phenomenon results from hosts being in the same protection domain and having the ability to start an attack from any other host in the same protection domain. This metric is sensitive to increase or decrease in exploitable vulnerabilities or vulnerable hosts. Because of this sensitivity, the Number of Paths metric can serve as a useful tool in monitoring the security of a network.

In the Flat network, this metric inherently weights the host designated as the target more weight than the other hosts in the network. In the Flat network, when the target host increases in vulnerabilities, its effect on the Number of Paths metric is greater than the effect of another host in the network increasing its exploitable vulnerabilities by a commensurate number of vulnerabilities. Table 5.1 shows the Number of Paths metric for different vulnerability distributions in the network. It can be observed from Table 5.1 that when new hosts stop joining the attack graph and the target host V increases in its number of vulnerabilities, no other host that increases by the same amount of vulnerabilities produces a larger change in the Number of Paths metric. This occurs because each host in the network is adjacent to V and therefore changes in V affect every path in the attack graph.

Table 5.1: Number of paths for different vulnerability distributions in the Flat network

Number of Vulnerabilities to Host Assignment	Number of Paths
1V	1
1B1V	2
1C1B1V	4
1D1C1B1V	8
1D1C1B2V	16
1D1C2B2V	24
1D2C2B2V	36
2D2C2B2V	54
2D2C2B3V	81
2D2C3B3V	108
2D3C3B3V	144
3D3C3B3V	192
3D3C3B4V	256
3D3C4B4V	320
3D4C4B4V	400
4D4C4B4V	500

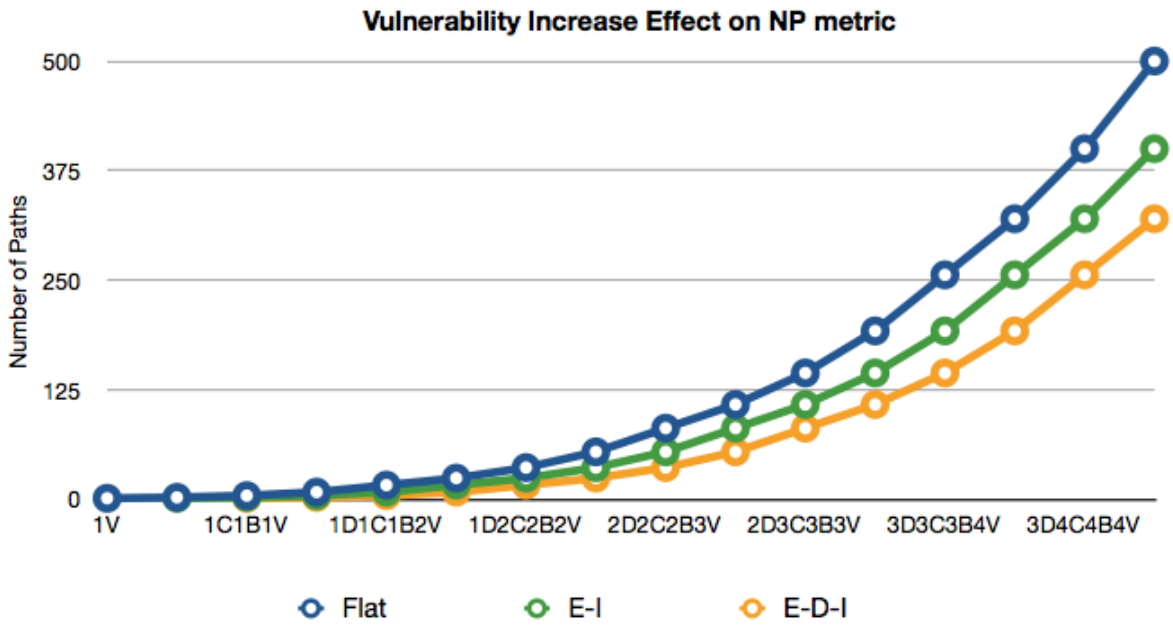

Fig. 5.6.: The effect of vulnerability increase on the number of paths metric under different network models

These findings suggest that new hosts that want to join a protection domain should be scrutinized more closely than hosts that have already been accepted into the protection domain. Looking at Table 5.1, the increase in the Number of Paths metric is most dramatic when a new host is discovered. When host B, C, and then D have a vulnerability first discovered, the Number of Paths metric increases 100% above its previous value.

When the E-I or E-D-I model is used, then the hosts that the attacker must use to reach the host have more dramatic effects on the networks security than the target machine. This result suggests that the target machine may not need as much fortification if its perimeter is secure. More specifically, the first host (from a dependency persepective) that the attacker can reach that has direct access to the victim host. By examining Table 5.2 and Table 5.3 we can observe that the path with the most dramatic effect is host B and host C respectively. Each of these hosts, in the E-I and E-D-I models respectively have the largest effect on the Number of Paths metric.

Table 5.2: Number of paths for different vulnerability distributions in the E-I network

Number of Vulnerabilities to Host Assignment	Number of Paths
1V	N/A
1B1V	1
1C1B1V	2
1D1C1B1V	4
1D1C1B2V	8
1D1C2B2V	16
1D2C2B2V	24
2D2C2B2V	36
2D2C2B3V	54
2D2C3B3V	81
2D3C3B3V	108
3D3C3B3V	144
3D3C3B4V	192
3D3C4B4V	256
3D4C4B4V	320
4D4C4B4V	400

Table 5.3: Number of paths for different vulnerability distributions in the E-D-I network

Number of Vulnerabilities to Host Assignment	Number of Paths
1V	N/A
1B1V	N/A
1C1B1V	1
1D1C1B1V	2
1D1C1B2V	4
1D1C2B2V	8
1D2C2B2V	16
2D2C2B2V	24
2D2C2B3V	36
2D2C3B3V	54
2D3C3B3V	81
3D3C3B3V	108
3D3C3B4V	144
3D3C4B4V	192
3D4C4B4V	256
4D4C4B4V	320

Characterizing the Number of Paths (NP) Metric Through experiments, we have extracted an equation to characterize the behavior the Number of Paths metric in response to changes in the number of vulnerabilities and hosts in the Flat network.

$$NP(G_t) = \begin{cases} v_t & \text{if } t = 1, \\ v_t NP(G_{t-1}) + NP(G_{t-1}) & t > 1. \end{cases} \tag{5.1}$$

The input parameter t corresponds to the event when a new host with a remotely exploitable vulnerability joins the network. The analysis assumes that once a machine joins the network with a given number of vulnerabilities, the number of vulnerabilities remains constant. If it changes then the equation would have to be recomputed. The proof of the above relation may be done for a family of general functions but not all general functions that the recurrence relation represents. The term v_t is actually a nondeterministic parameter. This parameter corresponds to the number of vulnerabilities that a host has when event t occurs, where $v_t > 0$. This flexibility makes this recurrence relation nondeterministic. Since nondeterministic functions cannot be described as a function of its inputs, we provide a proof for when the nondeterministic parameter is fixed to demonstrate the correctness of the above recurrence relation.

When $v_t = c$, we claim that $NP(G_t) = c(c+1)^{t-1}$ for $t \geq 1$.

Base case $t = 1$: $NP(G_1) = c(c+1)^{1-1}$

$= c(c+1)^0$

$= c(1)$

$= c$.

Base case holds because $v_t = c$.

The inductive hypothesis is that $NP(G_t) = c(c+1)^{t-1}$ for all $t \leq k$.

Prove claim holds for $t = k+1$.

$NP(G_{k+1}) = cNP(G_{k+1-1}) + NP(G_{k+1-1})$

$= cNP(G_k) + NP(G_k)$

$= c[c(c+1)^{k-1}] + c(c+1)^{k-1}$ by the inductive hypothesis

$= c(c+1)^{k-1}[c+1]$

$= c(c+1)^k$ QED.

The ramifications of this equation suggest that once an initial number of paths is known for an attack graph, we no longer need to recompute the entire attack graph in order to determine the Number of Paths metric. If the initial number of paths can be determined without an attack graph, then attack graph is not necessary for determining the Number of Paths metric. This computation scales linearly with respect to the number of hosts in the same protection domain.

There is alternative way of computing this metric for the different network models. Let n_{target} correspond to the number of remotely exploitable vulnerabilities at the *target* host. Let $n_{gateway_i}$ correspond to the number of remotely exploitable vulnerabilities at a host that serves as a gateway to the attacker for reaching a new protection domain. Let n_j correspond to the number of remotely exploitable vulnerabilities that exist at some host that is not the attacker's, the target's, or a gateway.

For the Flat network model, the Number of Paths metric can be computed as the following: $n_{target} \prod_{i=2}^{m} (n_i + 1)$, where m is the number of hosts in the network.

For the External-Internal network model, the Number of Paths metric can be computed as the following: $n_{target} \times n_{gateway_1} \prod_{i=3}^{m} (n_i + 1)$.

For the External-DMZ-Internal network model, the Number of Paths metric can be computed as the following: $n_{target} \times n_{gateway_1} \times n_{gateway_2} \prod_{i=4}^{m} (n_i + 1)$.

The External-Internal network model and the External-DMZ-Internal network model computation can be generalized as the following: $n_{target} \prod_{j=1}^{p} n_{gateway_j} \prod_{i=p+2}^{m} (n_i + 1)$, where p is the number of gateways between the attacker and the target.

Mean of Path Lengths (MPL) Metric Assessment

A salient issue with the Mean of Path Lengths metric is revealed in Figure 5.7. As the vulnerabilities in the network increases, the mean path increases. *The increase in path length indicates that the network's security is actually improving as a result of increasing the number of vulnerabilities in the network.* This interpretation is clearly

erroneneous. The MPL value increase is caused by increasing the number of circuitous routes an attacker can take to reach a target host.

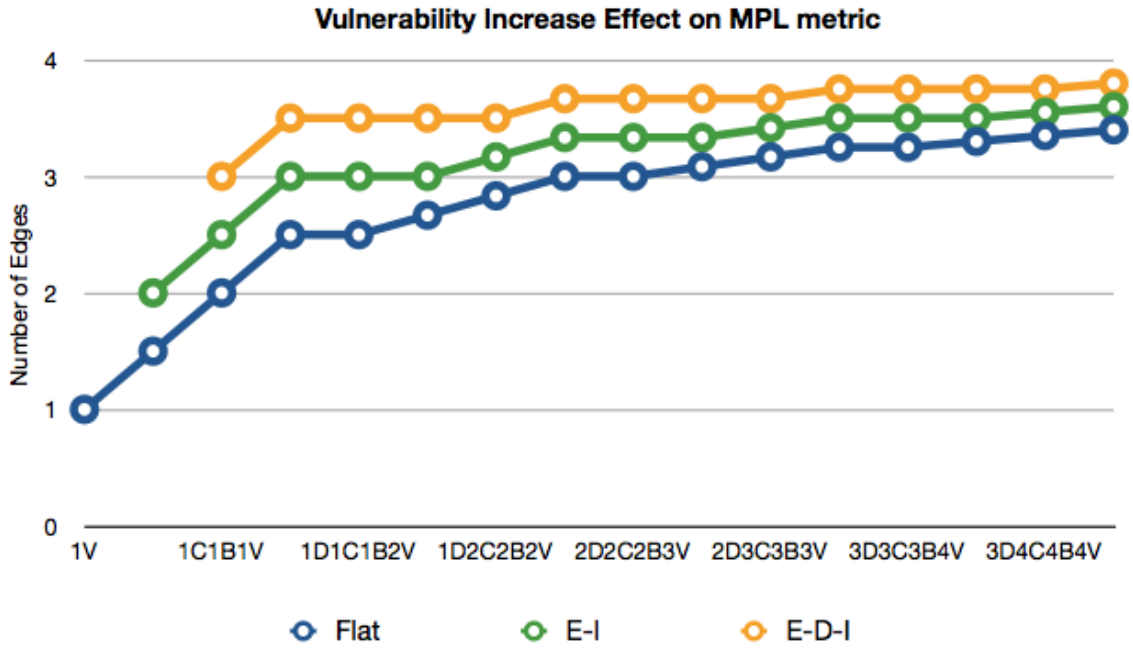

Fig. 5.7.: The effect of vulnerability increase on the mean path length metric under different network models

Although, there is no clear interpretation of the Mean of Path Lengths metric alone in Figure 5.7, computing this metric will be useful for computing the Normalized Mean of Path Length metric.

Characterizing the Mean of Path Lengths Metric

Let $z = MPL(G_{t-1})'(NP(G_t) - NP(G_{t-1}))$.

Let $y = MPL(G_{t-1})NP(G_{t-1})$.

$$MPL(G_t) = \begin{cases} l & \text{if } t = 1; \\ \frac{1}{NP(G_t)}[z + y] & \text{otherwise.} \end{cases} \tag{5.2}$$

Note that $MPL(G_{t-1})' = MPL(G_{t-1}) + l$, where l is the path length between two hosts on the same network. If all lengths between hosts are treated the same,

then the length will be treated as the unit length between two hosts. If the path length corresponds to some alternative function of complexity, then the path length will correspond to the average complexity associated with the vulnerabilities v_t.

Normalized Mean of Path Lengths (NMPL) Metric Assessment

Given our defintion of attack path length, the Mean of Path Lengths metric does not support the interpretation of the typical attack effort associated with an attack graph and for the reasons outlined in section 3.3.15, the Normalized Mean of Path Lengths metric (in Figure 5.8) is proposed. We see, as expected, that the Flat network model is the least secure network model. The External-Internal network model is the more secure than the Flat network model. The External-DMZ-Internal network model is more secure than the than External-Internal network model. However, as the network becomes more and more saturated with vulnerabilities, the difference in security levels of the different networks models become negligible.

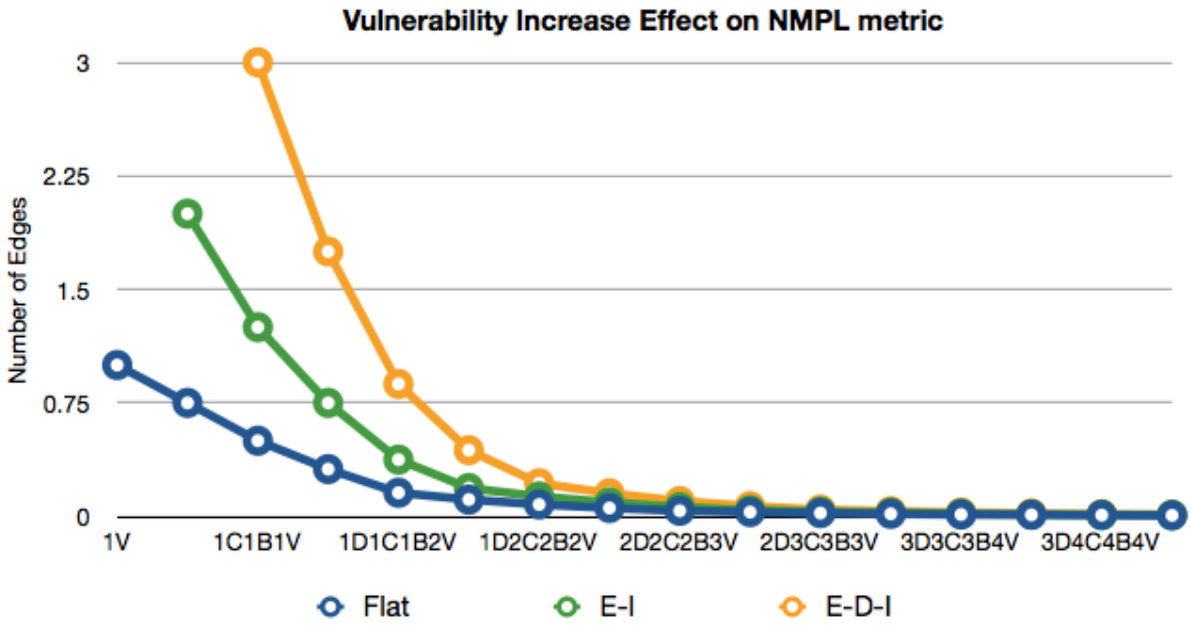

Fig. 5.8.: The effect of vulnerability increase on the normalized mean path length metric under different network models

Standard Deviation of Path Lengths (SDPL) Metric Assessment

Based on the data in Figure 5.9, we can see that the variability in attack path lengths increase more dramatically when new vulnerable hosts join the network. The variability in path length begins to decrease once new vulnerable hosts stop joining the network. However, as the vulnerabilities increase on the hosts already involved in some attack path, the variability of the path lengths decrease. This occurs because as the vulnerabilities on these hosts are incremented, the mean path length changes more slowly than when a new vulnerable host joins the network. This observation suggests that scrutinizing hosts attempting to join a network is warranted.

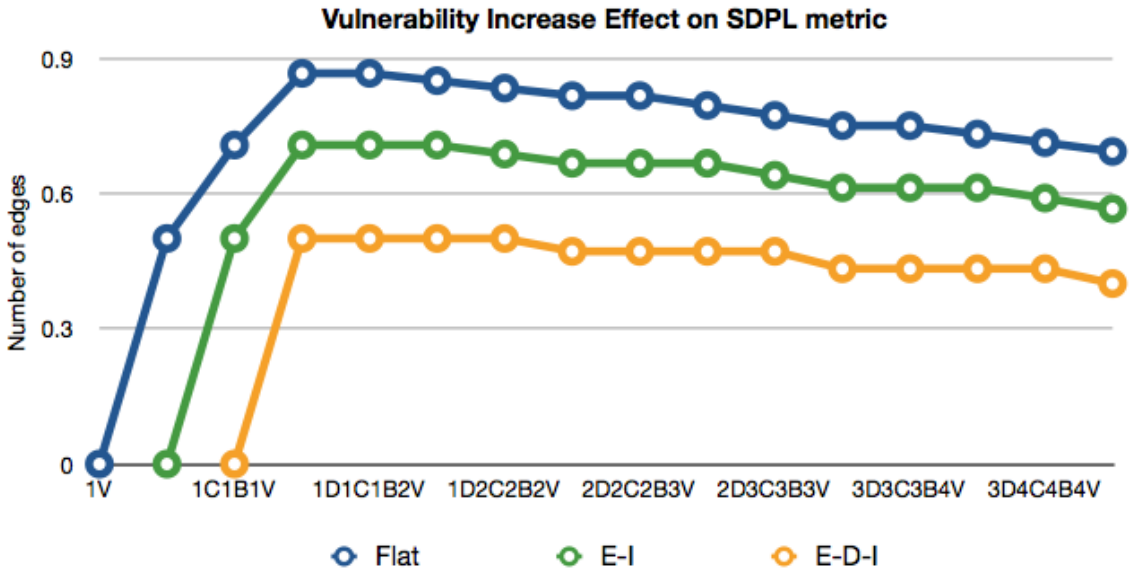

Fig. 5.9.: The effect of vulnerability increase on the standard deviation of path lengths metric under different network models

Although this data shows that increasing the access control of the network decreases variability in the path length, these values indicate little about the security of the system by themselves. Because these values cannot be used to decide which of two networks are most secure, this data supports our rationale for why we classify the Standard Deviation of Path Lengths metric as an Assistive metric. If desired,

this metric can be normalized in the same way the MPL metric can be normalized in order to be used with NMP instead of the MPL.

Median of Path Lengths (MePL) Metric Assessment

The chart in Figure 5.10 shows that there is a general trend of the median increasing. Given the results of the Standard Deviation of Path Lengths metric above, there is little surprise that there are more changes in the median in the Flat network than any other network, and that there are more changes in the median in the External-Internal network than in the External-DMZ-Network.

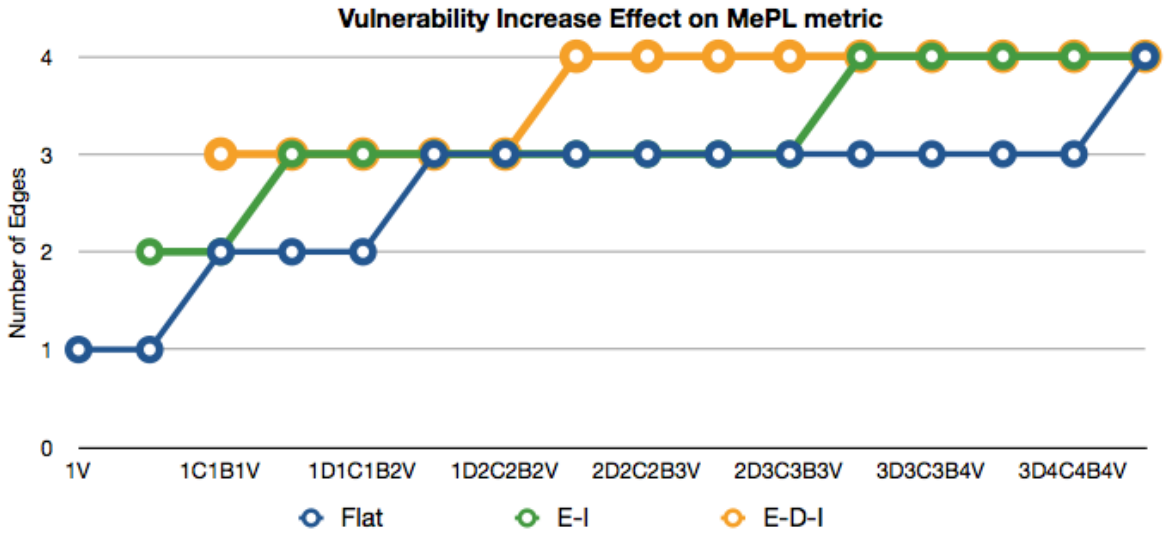

Fig. 5.10.: The effect of vulnerability increase on the median path length metric under different network models

The data shows why we do not classify the Median of Path Lengths metric as a decision metric. At various points in Figure 5.10, the Median of Path Lengths metric shows the different network models as being equally secure when they are actually not. This data, by itself, is insufficient to determine what network is most secure. As a tool for determining the security level of a network it provides a point of interest.

More specifically it provides the location of the 50th percentile, which may be useful in other anlayses (see section 4.2.4).

Mode of Path Lengths (MoPL) Metric Assessment

The Mode of Path Lengths metric in Figure 5.11 exemplifies why we do not classify the Mode of Path Lengths metric as a decision metric. With the exception of the flat network model, the mode stays constant as the number of vulnerabilities in the network increase and access controls are increased. By the flat network having some mode values that are below the other network models do suggest that the system is not as secure as the other network models. In order to use this is as a tool to determine the level of security of the system, one would need to know the frequencies associated with the mode(s). Thus, when comparing modes of two networks, an analyst would need to identify the network that corresponded to the mode with a lower frequency if the mode values are equal. The system with the lower mode frequencies would be considered more secure. Otherwise the analyst could take the higher of the two (or more) mode values. However, this approach cannot be relied on for deciding the security of a network, and is classified as an assistive metric.

Network Compromise Percentage (NCP) Metric Assessment

The NCP for the studied network models is shown in Figure 5.12. As can be seen, all that is required to compute this metric is knowledge of which hosts in the network are involved in some attack. Therefore, once there is at least one exploitable vulnerability on each host, the NCP will stop changing in value as it will be at its maximum value (100%). This reveals why this metric may not be useful in determining vulnerability changes in the network. This metric's level of granularity is at the host level. In Figure 5.12, as the number of vulnerabilities increase for any one host, NCP does not change. While this metric may be used to monitor the ongoing

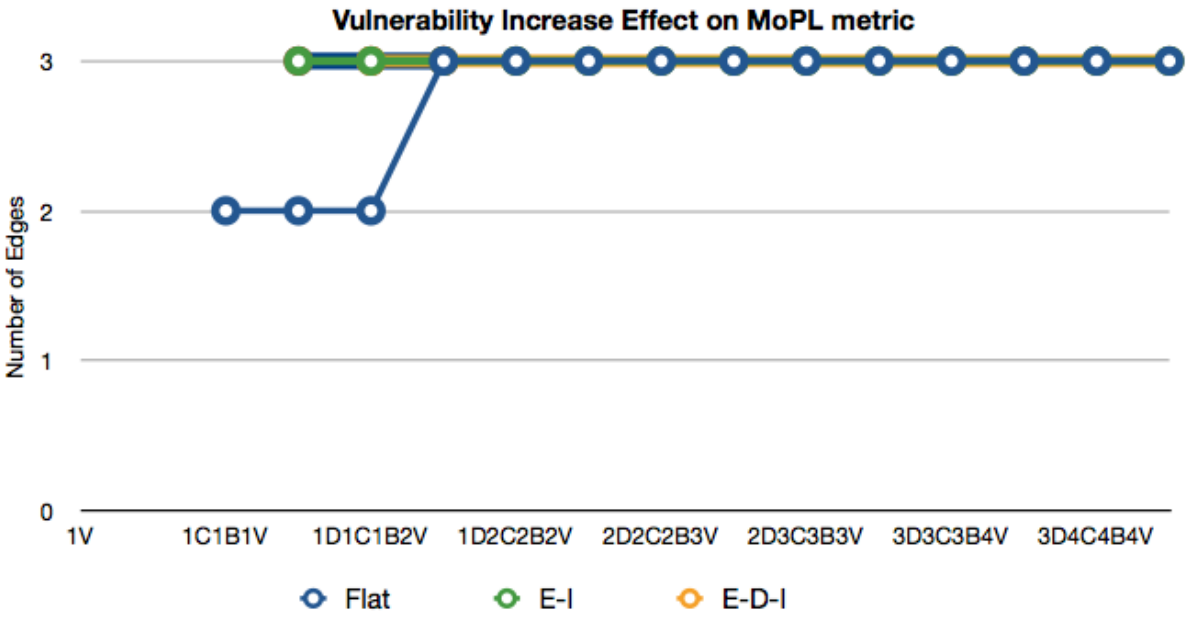

Fig. 5.11.: The effect of vulnerability increase on the mode path length metric under different network models

security of a network, its value depreciates if hosts can be exploited by more than one vulnerability.

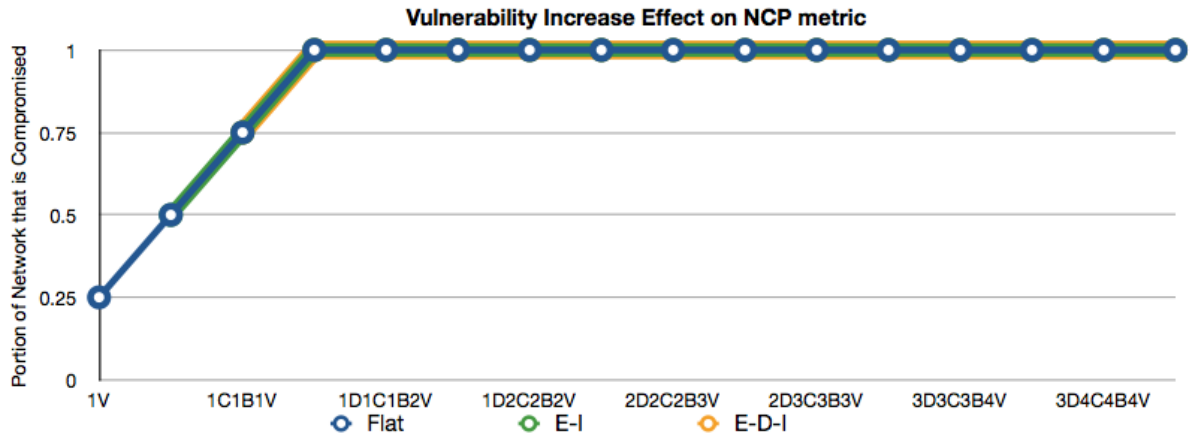

Fig. 5.12.: The effect of vulnerability increase on the network compromise percentage metric under different network models

5.2 Aggregation of Attack Graph-based Security Metrics Approach Evaluation

In this section, we show the results of using *compareGraphs* for two sets of randomly generated attack graphs. We generate this data randomly because obtaining real data would require organizations to divulge the ways attackers may be able to attack them. Even if the attack graph corresponds to a network configuration that is no longer valid, the attack graph may reveal intimate details about an organization's network that would be better kept confidential. First, we explain the experiment setup. Afterward, we present the results of performing our experiment.

5.2.1 Experiment Setup

The number of paths in the attack graphs are uniformly distributed between 1 and 2000 attack paths. The path lengths is uniformly distributed between 1 and 50. 1000 randomly generated attack graphs are assigned to set G_a, and a disjoint set of 1000 randomly generated attack graphs are assigned to set G_b. *compareGraphs* is called for the 1000000 pairs of attack graphs belonging to distinct sets. The value 2 is used for the parameter t for *compareGraphs*. The results are shown in Table 5.4.

5.2.2 Results

In Table 5.4, DMs and AMs refer to Decision Metrics and Assistive Metrics respectively. Table 5.4 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.

Table 5.4: The percentage of compared attack graphs for which *compareGraphs* reaches a decision

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

6. ATTACK GRAPH-BASED NETWORK HARDENING

In this chapter, we discuss how to use attack graph-based security metrics to protect a network with multiple options for countermeasures given a limited budget. We frame the problem as a combinatorial optimization problem and propose a dynamic programming solution. We exemplify how to use our method with a detailed example. We end this chapter with a specification of an approach for maximizing security across multiple security metrics.

6.1 Motivation

Network administrators fulfill the duty of preventing network attacks by identifying vulnerabilities in the network and then systematically removing the identified vulnerabilities. The removal of an identified vulnerability from a network may be referred to as a patch or a countermeasure.

A *countermeasure* is any action performed to remove at least one vulnerability from a system. Subsequently, the set of all countermeasures is infinite. However, practically, a network administrator will only consider a finite set of countermeasures for possible application to the network being protected. General countermeasures include, for example, modifying firewall rules, updating software on networked hosts, shutting down system services, or modifying an authentication routine.

The identification of vulnerabilities is critical to the effective use of countermeasures. A commonly used method to identify vulnerabilities is scanning the network for vulnerabilities via vulnerability scanners [65, 66]. A weakness of this method is that vulnerability scanners do not reveal the interdependencies that may exist between vulnerabilities found on different hosts of the same network. In order to compensate for this shortcoming, automated attack graphs can be used (e.g., [10]).

We purport that attack graph analyses providing network hardening recommendations could be enhanced to help network administrators be more effective at the countermeasures choosing problem (CCP). Informally, CCP is the following: given a limited budget, choose from a finite set of available countermeasures a subset of countermeasures that provide the highest security possible without going over budget. We propose to provide this analysis by modeling the CCP as a binary knapsack problem. We suggest the use of dynamic programming to solve the binary knapsack problem. Hence, our contribution includes:

- a novel approach for combining budget constraints and hardening recommendations into attack graph analysis
- specification of how security metrics can be used to choose hardening measures.

6.2 Related Work

In attack graphs, the application of countermeasures is simulated by removing some subset of vulnerabilities or exploits from its representation. The literature discussed in this section propose analyses that provide the network administrator with network hardening suggestions that, if implemented, produce a safe network or a more secure network with respect to a security metric.

In [58], Jha et al. attempt to find the smallest subset of measures that are needed to make the network safe. Jha et al. note that finding such a subset is equivalent to the minimum hitting set problem which is NP-complete [67]. The authors approximate a solution using a greedy approach where the measures preventing the most attacks are chosen in descending order. A drawback of this approach is that it is an approximation and yields potentially suboptimal solutions.

In [68], Noel et al. propose a minimum-cost network hardening method. Noel et al. propose the use of algebraic backwards substitution from an attack graph's goal state to its initial state. This backwards substitution yields the goal state in terms of the initial conditions. The boolean expression obtained for the initial conditions

is converted into conjunctive normal form yielding maxterms that are then evaluated on a lattice. Maxterms represent hardening suggestions that will preserve the safety of the network. Maxterms lower in the lattice correspond to hardening suggestions requiring the least cost or effort. The primary drawback of this approach is that it is binary. That is, the effectiveness of this approach hinges on the ability of the network administrator to implement all hardening suggestions for a given recommendation.

In [58, 68], the assumption is made that the network administrator has all the resources the network administrator needs to implement hardening recommendations. However, a network administrator's ability to safeguard a network is often times constrained by a limited budget [69]. Neither of the aforementioned approaches address this real domain challenge. Our approach deals with this challenge by incorporating the network administrator's funding constraint into the attack graph analysis to discover hardening recommendations.

Phillips and Swiler incorporate a budget into their attack graph analysis to generate hardening suggestions [9]. However, their algorithm follows a greedy approach that does not guarantee optimality. Furthermore, their analysis is based on knowing attacker costs or attacker success probabilities, which are difficult to ascertain in practice. Our approach guarantees optimality and does not rely on knowing attacker costs or attacker success probabilities.

In [55], Lippmann et al. describe a method for generating hardening recommendations that are derived from removing edges from the attack graph and observing its effect on the system's Network Compromise Percentage (NCP). A NCP of 0 percent would suggest a safe network. A NCP of 100 percent would suggest a network that is completely compromised. All hardening recommendations affects the NCP of the modeled system. When the analysis is done, the network administrator is presented with recommendations in ascending order of NCP. Although this approach takes large strides to assist the network administrator in making hardening decisions, the network administrator still has no assurance that the recommendations offered represent optimal usage of available resources.

Our approach compliments and enhances the above hardening recommendation method. Coupling our method with the one in [55] gives the network administrator the assurance that optimal recommendations are being received with respect to the network administrator's budget. We offer an algorithm for generating recommendations that are guaranteed to optimize network security with respect to a security metric (e.g., NCP) for the budget specified by the network administrator.

In [70], Chen et al. use the System Quality Requirements Engineering (SQUARE) methodology to perform a detailed case study. Although this work is not specifically for attack graph analysis, the work remains germane to our approach. The researchers were tasked with helping to secure a client's assets. The researchers used linear programming to determine the best set of countermeasures to choose given the budget their client allocated for security. The word "best" has a narrow definition in [70]. Best is defined as removing the most vulnerabilities from the network. However, this may not always be the case based on the metric being used.

Solving the problem of choosing countermeasures as a combinatorial optimization is consistent with our approach; however, there are important differences. Our method maintains all discovered optimal solutions, whereas a single optimal solution is provided in [70]. When considering hardening options, there are factors that are not easily parameterized that are better left to the judgment of the network administrator (e.g., difficulty of implementing certain countermeasures, long-term viability of a hardening recommendation). If a network administrator is presented with multiple optimal solutions, the network administrator can choose the best hardening recommendation based on past experience. Chen et al. use attack trees primarily for ancillary documentation purposes whereas in our approach attack graphs are integral. The network administrator can obtain a visual representation of the effect each countermeasure has on the attack graph and subsequently the network. These visual representations could serve as more effective documentation, communication, and forensic tools for the network administrator. Because our approach explicitly incorporates the use of security metrics, using a security metric such as the one sug-

gested in [25], our approach can capture the effect of making the exploitation of a particular vulnerability more difficult by removing some other vulnerability. The approach offered in [70] does not capture this form of vulnerability interdependence.

6.3 Maximizing Security Given a Limited Budget

The removal of security flaws is performed by implementing one or more countermeasures; however, the selection of the appropriate set of countermeasures is non-trivial. For example, discovering the "best" way of removing vulnerabilities could require the manual analysis of many combinations of countermeasures. Moreover, there may be overlap in the vulnerabilities that countermeasures remove. Let there be vulnerabilities v_1, v_2, v_3, v_4, v_5, and v_6 in a network *net*. There are the following countermeasures for *net*: c_1, c_2, and c_3. c_1 removes vulnerabilities v_1, v_5, and v_6. c_2 removes vulnerability v_1 and v_4, and c_3 removes v_1 and v_3. If only two countermeasures can be chosen of the three, which two countermeasures should be chosen? This issue not only exemplifies the difficulty of choosing countermeasures, but it highlights the importance of the security metric being used.

6.3.1 The Countermeasures Choosing Problem (CCP)

Due to the potentially many combinations of countermeasures to choose from and the countermeasures' possible overlap in protection, the selection of an optimal set of countermeasures is difficult. The problem of choosing the appropriate combination of countermeasures such that the security of the network is optimized and constrained to a given budget is called the Countermeasures Choosing Problem (CCP). The CCP formulation is inspired by the classic Binary Knapsack Problem.

The Binary Knapsack Problem

The Binary Knapsack Problem is a well-known optimization problem where the goal is to maximize a quantity subject to some constraint [71]. The problem can be formally defined as [72]: given a set of n items and a *knapsack* with

p_j = profit of item j,

w_j = weight of item j, and

c = capacity of *knapsack*,

select a subset of items so as to

maximize $z = \sum_{j=1}^{n} p_j x_j$

subject to $\sum_{j=1}^{n} w_j x_j \leq c$,

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is selected;} \\ 0 & \text{otherwise.} \end{cases}$$

The CCP Formulation

We model the CCP, similarly, as follows:

The countermeasures are labeled $1...n$.

m_j = security benefit of countermeasure j,

q_j = cost of implementing countermeasure j, and

B = budget of network administrator,

select a subset of hardening countermeasures so as to

maximize $v = \sum_{j=1}^{n} m_j x_j$

subject to $\sum_{j=1}^{n} q_j x_j \leq B$,

$x_j = 0$ or 1,

$$x_j = \begin{cases} 1 & \text{if countermeasure } j \text{ is selected;} \\ 0 & \text{otherwise.} \end{cases}$$

There are requirements associated with our model of the CCP. First, the model implicitly assumes that there exists a security metric to derive m_j. A countermeasure's m_j is calculated with respect to the countermeasures that are already being used. The rationale for this calculation constraint stems from the possible overlapping in the countermeasures' ability. For example, assume there are two countermeasures being considered. When either countermeasure is used in isolation, protection is provided to half the network. However, if both countermeasures are used at the same time, then it is not necessarily the case that the network is completely protected. The two countermeasures may overlap in what network assets they protect. Hence, m_j may take on different values depending on what countermeasures are already in place within the network. Our model also assumes that the network administrator is able to assign costs to the hardening measures in terms of money or time.

6.3.2 Solving the CCP

By giving the theory of dynamic programming, Bellman provided an exact solution for the binary knapsack problem [73]. We adopt the dynamic programming approach to solving the CCP. We define variables as the following:

Let R_l^j = maximum security possible using $x \subseteq \{1, 2, 3..., j\}$ which yields cost l;

$R_0^j = 0$;

$R_l^j = max\{R_l^{j-1}, R_{l-q_j}^{j-1} + m_j\}$;

$R_l^j = R_l^{j-1}$, if $R_l^{j-1}$is defined and $R_{l-q_j}^{j-1}$is undefined;

$R_l^j = R_{l-q_j}^{j-1} + m_j$, if $R_{l-q_j}^{j-1}$ is defined and R_l^{j-1} is undefined;

R_l^j = undefined, if R_l^{j-1} and $R_{l-q_j}^{j-1}$ are undefined.

The necessary steps to leverage our approach are: (1) determine the budget, (2) determine the security metric of interest, (3) generate the attack graph, (4) determine what countermeasures are available to safeguard the network and assign them costs, and (5) apply our dynamic programming algorithm to the inputs given above.

Step 1 will vary based on who within an organization sets the defense budget. Step 2 could be fulfilled by exploring the literature (e.g., [9, 25, 56, 58]). Step 3 could be achieved by using any of the tools or techniques discussed in [6, 10, 55, 64, 74] among other work. Step 4 could be accomplished by identifying all the exploits from the attack graph and associating each exploit with any countermeasure that removes the exploit from the attack graph. The cost categories specified in [70] could be used to come up with costs for the identified countermeasures. Algorithm 8 gives the dynamic programming algorithm that is to be applied in step 5.

The algorithm shown in Algorithm 8 maintains all discovered optimal solutions in R and AG, giving the network administrator the ability to choose from alternative optimal solutions. If there is an optimal value entry found in R there is a corresponding entry in AG. The *modify* function ensures that the countermeasure decision made by the algorithm is reflected in the corresponding attack graph in AG by removing the corresponding exploit edge(s) or node(s) depending on the type of attack graph being used. If V and E are the vertices and edges of the attack graph respectively, *modify*'s time complexity is $O(H)$ where $H = |V| + |E|$. The time complexity of ζ is contingent on the security metric chosen. However, if we assume that the security metric value can be obtained from a depth-first search of the attack graph (e.g., total number of attack paths), then the dynamic programming algorithm's time complexity is $O(nH^2B)$, otherwise the algorithm has a time complexity of $O(nHKB)$ where K is the time complexity of ζ. The countermeasures chosen for an optimal hardening recommendation can be determined by backtracking through R. We will give such an approach in section 6.3.3.

A Detailed Example

In this section, we will carry out an example on a network where 4 different countermeasures are being considered for application to the network. The cost of implementing countermeasures 1, 2, 3, and 4 is \$20K, \$20K, \$20K, and \$30K respec-

Algorithm 8 DP Algorithm for CCP

Require: n {the number of countermeasures under consideration} , B {the budget in money or time} , G {original unaltered attack graph under consideration} , AG {$(n+1) \times (B+1)$ attack graph entries} , R {$(n+1) \times (B+1)$ matrix} , S {the security benefit function}

1: initialize the zeroth row and column of R to $0s$
2: initialize the zeroth row and column of AG to Gs
3:
4: **for** $i \leftarrow 1$ to n **do**
5: **for** $b \leftarrow 0$ to B **do**
6: **if** $cost(i) \leq b$ **then**
7: $\delta \leftarrow b - cost(i)$
8: **if** $m_i \leftarrow S(AG, i, \delta) + R_\delta^{i-1} \geq R_b^{i-1}$ **then**
9: $R_b^i \leftarrow m_i + R_\delta^{i-1}$
10: $AG \leftarrow modify(AG, i, b, \delta))$
11: **else**
12: $R_b^i \leftarrow R_b^{i-1}$
13: $AG \leftarrow modify(AG, i, b)$
14: **end if**
15: **else**
16: $R_b^i \leftarrow R_b^{i-1}$
17: $AG \leftarrow modify(AG, i, b)$
18: **end if**
19: **end for**
20: **end for**

tively. In this example, there is no goal state for the attacker. What is of primary interest is what machines can be compromised by the attacker. Based on this flexibility, we choose to use the predictive attack graph. Because the predictive attack graph may only have certain attack graph-based security metrics applied to it, we apply the NCP metric. When storing the values obtained from the NCP we subtract it from 100% to attain the values used in the R matrix. We also assume that the connectivity of the hosts in the network is static–that is, hosts are not leaving and joining autonomously.

Figures 6.1(a) and 6.1(b) correspond to the R matrix and the AG matrix when first initialized. The attack graphs in the AG matrix correspond to the attack graph that was initially generated for the network under inspection.

The algorithm finds that countermeasure 1 becomes affordable at $20K. This finding is depicted in Figure 6.2. The corresponding value is derived from applying countermeasure 1 to the network such that host E's vulnerability can no longer be exploited. The directed edges identify the cells that are responsible for values produced at the heads of the edges.

Similarly, the algorithm finds that countermeasure 2 becomes affordable at $20K. This finding is depicted in Figure 6.3. Countermeasure 2 hardens the network such that the vulnerability at host D can no longer be exploited. Once the countermeasures are considered for a budget of $40K, the benefit of using countermeasures 1 and 2 is captured by the matrices.

The final R and AG matrices in Figure 6.4 show the result of evaluating countermeasures 3 and 4. By examining the AG matrix, we can observe that countermeasure 3 provides benefit equivalent to countermeasure 2. Similarly, countermeasure 4 provides protection equivalent to countermeasure 1. The optimal attack graph for the given set of countermeasures can be found in cells (2,4), (2,5), (3,4), (3,5), (4,4), and (4,5). Algorithm 8 ensures that an optimal attack graph appears in the right-bottom-most cell.

	$0	$10K	$20K	$30K	$40K	$50K
0	0%	0%	0%	0%	0%	0%
1	0%					
2	0%					
3	0%					
4	0%					

(a) Initial R matrix

	$0	$10K	$20K	$30K	$40K	$50K
0						
1						
2						
3						
4						

(b) Initial attack graph matrix

Fig. 6.1.: Initial matrices

6.3.3 Finding All Optimal Countermeasures

In this section, we detail our algorithm for finding all optimal countermeasures. After detailing this algorithm, we specify the result of applying this algorithm to Figure 6.4(a).

	$0	$10K	$20K	$30K	$40K	$50K
0	0%	0%	0%	0%	0%	0%
1	0%	0%	20%	20%	20%	20%
2	0%					
3	0%					
4	0%					

(a) The result of applying countermeasure "1" in the R Matrix

	$0	$10K	$20K	$30K	$40K	$50K
0						
1						
2						
3						
4						

(b) The result of applying countermeasure "1" in the AG Matrix

Fig. 6.2.: Matrices after considering countermeasure "1"

Algorithm for Finding All Optimal Countermeasures

Algorithm 9 begins by setting the indices i and j to refer to the right-most-bottom corner of the R matrix. S is the Stack abstract data type. C is the Collection abstract data type. Collections may hold multiple lists.

The cell in the right-most-bottom corner of R will be the first cell to be evaluated once in the while loop (line 6). The first if statement determines whether multiple

	$0	$10K	$20K	$30K	$40K	$50K
0	0%	0%	0%	0%	0%	0%
1	0%	0%	20%	20%	20%	20%
2	0%	0%	20%	20%	40%	40%
3	0%					
4	0%					

(a) The result of considering applying countermeasures "2" in the R Matrix

	$0	$10K	$20K	$30K	$40K	$50K
0						
1						
2						
3						
4						

(b) The result of considering applying countermeasure "2" in the AG Matrix

Fig. 6.3.: Matrices after considering countermeasure "2"

optimal sets exists that can produce R_{ij}. The "matches" relation is to be interpreted as testing whether the arithmetic difference between its two operand values is equal to i's security benefit. When matches produces true, then i is in some optimal list of countermeasures. If matches is false (or "does not match" is true) then i is not in some optimal list of countermeasures.

Algorithm 9 Finding All Optimal Countermeasures Algorithm

Require: R $\{(n+1) \times (B+1)$ matrix of R values$\}$

1: $i \leftarrow n+1$
2: $j \leftarrow B+1$
3: Stack S
4: Collection C
5: $S.push(R_{ij})$
6: **while** i is not equal to 0 **do**
7: $\quad \delta \leftarrow cost(i)$
8: $\quad$ **if** R_{ij} matches $R_{i-1\delta}$ and R_{ij} does not match R_{i-1j} **then**
9: $\quad\quad C' \leftarrow copy(C)$
10: $\quad\quad C'.addCountermeasureToAll(i)$
11: $\quad\quad C.combine(C')$
12: $\quad\quad S.pop()$
13: $\quad\quad S.push(R_{i-1j})$
14: $\quad\quad S.push(R_{i-1\delta})$
15: $\quad$ **else if** R_{ij} matches $R_{i-1\delta}$ **then**
16: $\quad\quad C.addCountermeasureToAll(i)$
17: $\quad\quad S.pop()$
18: $\quad\quad S.push(R_{i-1\delta})$
19: $\quad$ **else if** R_{ij} matches R_{i-1j} **then**
20: $\quad\quad S.pop()$
21: $\quad\quad S.push(R_{i-1j})$
22: $\quad$ **end if**
23: $\quad$ **if** j equals 0 and i does not equal 0 **then**
24: $\quad\quad S.pop()$
25: $\quad$ **end if**
26: **end while**
27: **return** $prune(C)$

	$0	$10K	$20K	$30K	$40K	$50K
0	0%	0%	0%	0%	0%	0%
1	0%	0%	20%	20%	20%	20%
2	0%	0%	20%	20%	40%	40%
3	0%	0%	20%	20%	40%	40%
4	0%	0%	20%	20%	40%	40%

(a) The R Matrix with all of its values computed

	$0	$10K	$20K	$30K	$40K	$50K
0						
1						
2						
3						
4						

(b) The AG Matrix with all of its graphs computed

Fig. 6.4.: Matrices after considering countermeasures “3” and “4”

When the first if statement in Algorithm 9 is true, two collections are created. One collection will include countermeasure i in all of its list, and the other collection will not include countermeasure i in all of its lists. The union of these two lists are obtained to create a new single collection C. The next cells to be evaluated are pushed onto the stack after removing the R_{ij} cell. The second if statement (line 15) tests whether the current countermeasure is apart of all optimal lists. The last else-if statement

(line 19) tests whether the algorithm should discard the current countermeasure and advance to the next countermeasure. The *addCountermeasureToAll* method only adds a countermeasure i if adding countermeasure i does not exceed the budget. The method *prune* removes any lists that are suboptimal (i.e., does not provide maximal security benefit). When this algorithm is applied to the Figure 6.4(a), all distinct pairs of two countermeasures is produced. Because this algorithm finds all optimal countermeasures and there can n choose 2 of them at most, this algorithm is $O(n!)$

6.4 Optimizing Security for Multiple Metrics

To incorporate multiple metrics in our network hardening approach, we utilize an Aggregated Objective Function (AOF). The AOF consists of all attack graph-based security metrics a network administrator may be interested in using. The network administrator may use any subset of attack graph-based security metrics discussed in this dissertation. We now provide equations that translate each metric into the same unit. Once the metrics are in the same unit, a network administrator can weight the metrics as desired.

6.4.1 Attack Graph-based Security Metric Translation

In this section we translate decision metrics so that they are of the same unit. We create translations so that larger values correspond to better security. Let G correspond to an attack graph in the equations below. We append the subscript r to each metric to signify that it is the translated version of the original security metric.

Shortest Path (SP) Metric Translation

$SP(G)$ produces the shortest path in attack graph G. A shorter attack path correlates to weaker security. Thus, as $SP(G)$ increases the security of the system should be improving. These semantics are captured in the equation below.

$SP(G)_r = \frac{SP(G)}{maxLength(G)}$

$maxLength(G)$ is a function that returns the longest path in attack graph G. Provided that $maxLength(G)$ is satisfactorily long, $SP(G)_r$ can provide a relevant indicator of the network's security with respect to the $SP(G)$.

Number of Paths (NP) Metric Translation

$NP(G)$ produces higher values when the security of the network is decreasing. Thus, the inverse of this value produces the desired effect of a value between 1 and 0, with better security being closer to 1 than 0.

$NP(G)_r = NP(G)^{-1}$

Normalized Mean of Path Lengths (NMPL) Metric Translation

$NMPL(G)$ represents the typical effort an attacker can expend given the number of attack paths in the network. Thus, to obtain a value on a scale we desire, we normalize $NMPL(G)$ by the product of the $maxLength(G)$ and $NP(G)_r$.

$$NMPL(G)_r = \begin{cases} \frac{NMPL(G)}{maxLength(G) \times NP(G)_r} & \text{if } NMPL(G) \text{ and } maxLength(G) \neq 1 \\ 0 & \text{otherwise} \end{cases}$$

If $maxlength(G)$ is 1 and $NMPL(G)$ is also 1, then a misleading value can be produced by $NMPL(G)_r$. Thus, we handle this case by inversing the result obtained from the general case.

Network Compromise Percentage (NCP) Metric Translation

$NCP(G)$ represents the percent of hosts that are compromised in a network. As this metric increases, the network security of the system decreases. Thus, to translate this metric, we transform the metric from a percentage to a fraction and subtract it from 1 to discover the assets that have not been compromised by the attacker.

$NCP(G)_r = 1 - \frac{NCP(G)}{100}$

Weakest Adversary (WA) Metric Translation

$WA(G)$ purports that the security of a network corresponds to the capabilities required to violate a security policy. Thus, the system is as secure as the weakest set of attacker capabilities needed to violate a security policy. Assuming the attacker attributes to be equally weighted, the translation for this metric is given below.

$WA(G)_r = \frac{C}{M}$

C corresponds the weakest set of attacker capabilities required to violate a security policy. M corresponds the total attacker capabilities considered during attack graph generation. Thus, under this interpretation, the more capabilities that an attacker requires to violate a network's security policy, the better the network's security is with respect to $WA(G)_r$.

Attack Resistance (AR) Metric Translation

$AR(G)$ produces higher values when network security improves and lower values when the network security degrades. Thus, to translate this metric, we subtract the inverse of $AR(G)$ from 1. The equation for the translation is given below.

$AR(G)_r = 1 - AR(G)^{-1}$

K-step Capability Accumulation Metric Translation

The K-step Capability Accumulation metric states that if an attacker can accumulate more capabilities in a network in a few steps, then the security of the network is poor. However, if the attacker requires more steps to obtain the same number of capabilities then the associated network is stronger than the aforementioned network. Thus, we are interested in identifying what percentage of capabilities the attacker is unable to attain. This notion is captured in the translation below.

$KCA(G)_r = 1 - \frac{C}{M}$, where C corresponds to the capabilities that the attacker can obtain, and M corresponds to the total number of capabilities in the network.

6.4.2 The Aggregated Objective Function

The objective function may include use any subset of decision metrics the network administrator deems useful for the evaluation. Let f correspond to the aggregated objective function. Let w_i correspond to a weight that the network administrator associates with a security metric. Then the aggregated objective function is given by the following equation.

$m_j = w_1 SP(G^{j-1}_{l-q_j})_r + w_2 NP(G^{j-1}_{l-q_j})_r + w_3 NMPL(G^{j-1}_{l-q_j})_r + w_4 NCP(G^{j-1}_{l-q_j})_r + w_5 WA(G^{j-1}_{l-q_j})_r + w_6 AR(G^{j-1}_{l-q_j})_r + w_7 KCA(G^{j-1}_{l-q_j})_r$, where j corresponds to countermeasure j, and q_j corresponds to the cost of implementing countermeasure j.

This AOF may be used within our objective function $S(AG, i, \delta)$ in our dynamic programming approach specified in Algorithm 8.

7. CONCLUSION AND FUTURE WORK

In this dissertation, we have examined a specific class of security metrics called attack graph-based security metrics. In describing the derivation of these metrics, we have identified the limitations of this class of security metrics. We have proposed a set of attack graph-based security metrics to compliment existing attack graph-based security metrics. We have provided a categorization scheme for attack graph-based security metrics that prescribe how these metrics should be applied independently and in combination in evaluating network security.

We conducted two simulation studies. In our first simulation study, we provided evidence that supports our classification of attack graph-based security metrics. This simulation study also provided insights regarding how and when these security metrics should be used to evaluate network security. From this simulation study, we discovered equations for two previously proposed attack graph-based security metrics: the Number of Paths metric, and the Mean Path Length metric. Equations for these security metrics provides another equation for a security metric we propose in this dissertation: the Normalized Mean Path metric.

In our second simulation study, we evaluated our algorithm for network security evaluation of two networks using synthetic attack graphs. The simulation showed that our approach for combining attack graph-based security metrics reaches a decision in determining the most secure network (i.e., least non-compliant network) between two networks in most cases. We assert that because our attack graphs were generated independently of an underlying network, our results hold for any pair of networks that generate attack graphs where the number of paths in the attack graph greatly outnumber the attack path lengths.

In stating where attack graph-based security metrics can be applied in an organization's security program, this dissertation has removed a barrier to adopting

these metrics in the practice. Also, in providing the limitations of these security metrics, this dissertation provides a guide that will help prevent security analysts from inappropriately using attack graph-based security metrics.

By providing equations for the Number of Paths metric, Mean Path Length metric, and subsequently the Normalized Mean Path Length metric, this dissertation has provided an efficient mechanism for computing these metrics by avoiding traversing the attack graph to derive these metrics. Such efficient computation of these metrics can make them viable for real-time monitoring of a network's security.

Lastly, we have provided an algorithm for network hardening recommendations. We model and solve the problem of providing network hardening recommendations as a combinatorial optimization problem. We have also shown how to combine metrics into an aggregated objective function for solving this optimization problem. With this approach, an organization can ensure that, with respect to the attack graph-based security metrics used, it has maximized its security given the budget it has allocated to hardening its network.

Future work consists of developing a formalized model for applying the attack graph to individual hosts. There is nothing inherent in attack graphs that make them exclusively useful for networks. Providing such a model has the important benefit of making evaluation of attack graph-based security metrics on real systems easier to accomplish. By assessing a single host, researchers would have the ability to assess how well different attack graph-based security metrics correspond to security incidents without putting in the effort to build an entire network.

LIST OF REFERENCES

LIST OF REFERENCES

[1] I. R. Council, "Hard problem list," Technical Report, DHS Cyber Security Research and Development Center, November 2005.

[2] A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley, Pearson Education, 2007.

[3] N. Bartol, B. Bates, K. Goertzel, and T. Winograd, "Measuring cyber security and information assurance: State-Of-The-Art Report (SOAR)," Technical Report, Information Assurance Technology Analysis Center (IATAC), May 2009.

[4] M. Bishop, *Computer Security*. Addison-Wesley, 2002.

[5] E. Schneider, "Measurements of system security," in *Proceedings of the 1st Information Security System Rating and Ranking Workshop (ISSRR '01)*, May 2001.

[6] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, November 2002.

[7] R. P. Lippmann, K. W. Ingols, C. Scott, K. Piwowarski, K. J. Kratkiewicz, M. Artz, and R. K. Cunningham, "Evaluating and strengthening enterprise network security using attack graphs," Technical Report, MIT Lincoln Laboratory, 2005.

[8] P. Ammann, J. Pamula, R. Ritchey, and J. Street, "A host-based approach to network attack chaining analysis," in *Proceedings of the 21st Annual Computer Security Applications Conference (ASAC '05)*, December 2005.

[9] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *NSPW '98: Proceedings of the 1998 Workshop on New Security Paradigms*, (New York, NY, USA), pp. 71–79, ACM, 1998.

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

[11] R. Lippmann and K. Ingols, "An annotated review of past papers on attack graphs.," Technical Report, MIT Lincoln Laboratory, March 2005.

[12] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 109–118, 2004.

[13] S. Braynov and M. Jadiwala, "Representation and analysis of coordinated attacks," in *Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering*, pp. 43–51, 2003.

[14] T. Garfinkel, "Traps and pitfalls: Practical problems in system call interposition based security tools.," in *Proceedings of the Network and Distributed Systems Security Symposium*, February 2003.

[15] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proceedings of Computer Security Applications Conference*, pp. 121–130, December 2006.

[16] X. Ou, W. Boyer, and M. McQueen, "A scalable approach to attack graph generation," *ACM Conference on Computer and Communications Security (CCS '06)*, November 2006.

[17] D. Herrmann, *A Practical Guide to Security Engineering and Information Assurance*. Auerbach Publications, 2002.

[18] N. Fenton and S. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, 1997.

[19] N. Fenton, R. Whitty, and Y. Iizuka, *Software Quality Assurance and Measurement: A Worldwide Perspective*. International Thomson Computer Press, 1995.

[20] System Security Engineering Capability Maturity Model (SSE-CMM), "http://www.sse-cmm.org/metric/metric.asp," 2010.

[21] D. Herrmann, *Complete Guide to Security and Privacy Metrics*. Auerbach Publications, Talyor and Francis Group, 2007.

[22] Mitre Common Vulnerabilities and Exposures (CVE), *http://www.cve.mitre.org*, July 2010.

[23] Common Criteria for Information Technology Security Evaluation, *http://www.commoncriteriaportal.org/thecc.html*, September 2007.

[24] Center for Education and Research in Information Assurance and Security, McAfee, *http://resources.mcafee.com/content/NAUnsecuredEconomiesReport*, 2009.

[25] 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, August 2007.

[26] R. Dantu and P. Kolan, "Risk management using behavior based bayesian networks," *Intelligence and Security Informatics, LNCS*, vol. 3495, pp. 115–126, 2005.

[27] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*. John Wiley and Sons, 2000.

[28] D. Redelmeier, "The cognitive psychology of missed diagnoses," *Annals of Internal Medicine*, vol. 142, pp. 115–120, January 2005.

[29] Bugtraq, *http://www.securityfocus.com/archive/1*, July 2010.

[30] Computer Emergency Response Team (CERT), "http://www.cert.org/," 2009.

[31] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security and Privacy*, vol. 4, pp. 85–89, 2006.

[32] IEEE Computer Society 1990, *Standard glossary of software engineering terminology, ANSI/IEEE Standard 610.12-1990*, December 1990.

[33] International Electrotechnical Commission (IEC), *IEC 615-7, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related System – Part 7 Overview of Techniques and Measures*, 1998.

[34] J. Alves-Foss and S. Barbosa, "Assessing computer security vulnerability," *ACM SIGOPS Operating Systems Review*, pp. 3–13, July 1995.

[35] E. Jonsson and T. Olovsson, "A quantitative model of the security intrusion process based on attacker behavior," *IEEE Transactions on Software Engineering*, April 1997.

[36] J. McDermott, "Attack-potential-based survivability modeling for high-consequence systems," in *Proceedings of the Third IEEE International Workshop on Information Assurance*, 2005.

[37] G. Schudel and B. Wood, "Adversary work factor as a metric for information assurance," in *Proceedings of the 2000 Workshop on New Security Paradigms*, pp. 23–30, 2001.

[38] B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K. Trivedi, "Modeling and quantification of security attributes of software systems," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN '02)*, 2002.

[39] P. O'Connor, *Practical Reliability Engineering*. John Wiley and Sons, Ltd., 1991.

[40] K. Sallhammar, B. Helvik, and S. Knapskog, "On stochastic modeling for integrated security and dependability evaluation," *Journal of Networks*, vol. 1, 2006.

[41] H. Hecht, *Systems Reliability and Failure Prevention*. Artech House, 2004.

[42] M. Y. Liu and I. Traore, "Empirical relation between coupling and attackability in software systems:: a case study on DoS," in *Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security (PLAS '06)*, (New York, NY, USA), pp. 57–64, ACM Press, 2006.

[43] H. Langweg, "Framework for malware resistance metrics," in *Proceedings of the Quality of Protection 2006 (QoP '06)*, October 2006.

[44] M. Abedin, S. Nessa, E. Al-Shaer, and L. Khan, "Vulnerability analysis for evaluating quality of protection of security policies," in *Proceedings of Quality of Protection 2006 (QoP '06)*, October 2006.

[45] M. Ahmed, E. Al-Shaer, and L. Khan, "A novel quantitative approach for measuring network security," in *Proceedings of IEEE INFOCOM 2008*, April 2008.

[46] NIST's National Vulnerability Database (NVD), "http://nvd.nist.gov," 2009.

[47] M. Howard, "Fending off future attacks by reducing attack surface," Technical Report, Microsoft Corporation, February 2003.

[48] M. Howard, J. Pincus, and J. M. Wing, "Measuring relative attack surfaces," in *Proceedings of Workshop on Advanced Developments in Software and Systems Security*, 2003.

[49] P. Manadhata and J. M. Wing, "Measuring a systems attack surface," Technical Report CMU-CS-04-102, Carnegie Mellon, Janurary 2004.

[50] P. Manadhata and J. M. Wing, "An attack surface metric," Technical Report CMU-CS-05-155, Carnegie Mellon, July 2005.

[51] P. Manadhata, Y. Karabulut, and J. Wing, "Measuring the attack surfaces of sap business applications," in *Proceedings of IEEE International Symposium on Software Reliability Engineering (ISSRE '08)*, 2008.

[52] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, pp. 633–650, September 1999.

[53] W. Li and R. Vaughn, "Cluster security research involving the modeling of network exploitations using exploitation graphs," in *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and Grid Workshops*, May 2006.

[54] S. Noel, M. Jacobs, P. Kalapa, and S. Jajodia, "Multiple coordinated views for network attack graphs," in *Proceedings of IEEE Workshop on Visualization for Computer Security*, pp. 99–106, 2005.

[55] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham, "Validating and restoring defense in depth using attack graphs," in *Military Communications Conference*, October 2006.

[56] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A weakest-adversary security metric for network configuration security analysis," in *Proceedings of the 2nd ACM Workshop on Quality of Protection (QoP '06)*, pp. 31–38, 2006.

[57] L. Wang, A. Singhal, and S. Jajodia, "Toward measuring network security using attack graphs," in *Proceedings of the 2007 ACM workshop on Quality Protection*, pp. 49–54, 2007.

[58] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, June 2002.

[59] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," *DAS 2008, LNCS 5094*, pp. 283–296, 2008.

[60] D. Balzarotti, M. Monga, and S. Sicari, "Assessing the risk of using vulnerable components," in *Proceedings of the 2nd ACM Workshop on Quality of Protection*, 2005.

[61] L. Ming and P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

[62] S. Evans, S. Bush, and J. Hershy, "Information assurance through kolmogorov complexity," *DARPA Information Survivability Conference and Exposition II (DISCEX-II-2001)*, June 2001.

[63] 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.

[64] X. Ou, S. Govindavajhala, and A. Appel, "Mulval: a logic-based network security analyzer," in *Proceedings of the 14th conference on USENIX Security Symposium*, vol. 14, 2005.

[65] Nessus Security Scanner, "http://www.nessus.org," 2008.

[66] Network Mapper (Nmap), "http://www.nmap.org," 2008.

[67] M. Garey and D. Johnson, *Computers and Intractibility*, ch. SP8. W.H. Freeman and Company, 1979.

[68] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs, "Efficient minimum-cost network hardening via exploit dependency graphs," in *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC '03)*, 2003.

[69] L. Bodin, L. Gordon, and M. Loeb, "Evaluating information security investments using the analytic hierarchy process," *Communications of the ACM*, vol. 48, February 2005.

[70] P. Chen, M. Dean, D. Ojoko-Adams, H. Osman, L. Lopez, and N. Xie, "Systems quality requirements engineering (square) methodology: Case study on asset management system," Technical Report, Carnegie Mellon University/Software Engineering Institute, 2004.

[71] United States National Institute of Standards and Technology (NIST), *http://www.nist.gov/dads/HTML/01KnapsackProblem.html*, July 2008.

[72] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementation*, ch. 2. John Wiley & Sons, 1990.

[73] R. Bellman, "The theory of dynamic programming," in *Proceedings of the National Academy of Sciences*, pp. 716–719, 1952.

[74] B. Schneier, "Modeling security threats," *Dr. Dobb's Journal*, December 1999.

VITA

VITA

Nwokedi C. Idika

Education

- Ph.D. Computer Science, Purdue University, 2010
- M.S. Computer Science, Purdue University, 2007
- B.S. Computer Science, University of Maryland Baltimore County, 2005

Employment

- Purdue University, Research Assistant, 2010
- Purdue University, Teaching Assistant, 2008-2010
- MITRE Corporation, Software Engineer, 2004-2006 (3 Summers, 1 Semester)
- Yale University, Research Fellow, 2003
- Army Research Lab, Technical Intern, 2002

Publications

- N. Idika and B. Bhargava, "Extending Attack Graph-based Security Metrics and Aggregating Their Application," IEEE Transactions on Dependable and Secure Computing (to appear).
- N. Idika, B. Marshall, and Bhargava, B., "Maximizing Security Given a Limited Budget," The Fifth Tapia Celebration of Diversity in Computing Conference, 2009.

- B. Bhargava, Y. Zhang, N. Idika, L. Lilien and M. Azarmi, "Collaborative Attacks in WiMAX Networks," Wiley's Security and Communication Networks Journal Special Issue on: WiMAX Security and Applications, 2009.
- R. Oliveira, B. Bhargava, Y. Zhang, and N. Idika, "Collaborative Attacks & Defense in Wireless Networks," International Conference on Distributed Computing, 2009.
- N. Idika and A. Mathur, "Survey of Malware Detection Techniques," Software Engineering Research Center, 2007.

Honors & Awards

- Applied Management Principles Certificate, *Purdue University's Krannert School of Management*
- Department of Homeland Security (DHS) S.T.E.M. Scholar, *Purdue University & DHS*
- Graduate Student Award for Outstanding Teaching, *Purdue University*
- Upsilon Pi Epsilon, *UMBC, Purdue University*
- Committee on Institutional Cooperation and General Electric (CIC/GE) Fellow, *CIC & GE*
- Purdue Doctoral Fellow, *Purdue University*
- Summa cum Laude, *University of Maryland Baltimore County*
- Meyerhoff Scholar, *University of Maryland Baltimore County*
- Phi Beta Kappa *University of Maryland Baltimore County*