A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks[*]

John Homer
Kansas State University
Manhattan, KS
jhomer@ksu.edu

Xinming Ou
Kansas State University
Manhattan, KS
xou@ksu.edu

David Schmidt
Kansas State University
Manhattan, KS
das@ksu.edu

Abstract

Mitigation of security risk is an important task in enterprise network security management. However it is presently a skill acquired by individual experience, more an art than a science. The biggest challenge in the problem is a quantitative model that objectively measures the likelihood a breach can be accomplished. This paper presents a sound and practical approach to such a quantitative model. We utilize existing work in attack graphs and individual vulnerability metrics, such as CVSS, and apply probabilistic reasoning to produce a sound risk measurement. The problem requires a careful coordination of attack graph data to account for cyclic and shared dependencies. We recognize that networks commonly have many host interconnections and network privileges could be gained in many ways. This factor leads to cycles in an attack graph, which must be identified and properly treated when measuring risk to prevent distortion of the results. We also recognize that multiple attack paths leading to the same network privilege will often share some dependencies and so a valid assessment cannot simply treat these paths as independent. Our approach is provably sound and ensures that shared dependencies have a proportional effect on the final calculation, and that cycles are handled correctly so that privileges are evaluated without any self-referencing effect. We also present preliminary experimental results on our algorithm and identify directions for future improvement.

1 Introduction

Enterprise networks have become essential to the operation of companies, laboratories, universities, and government agencies. As they continue to grow both in size and complexity, network security has become a critical concern. Vulnerabilities are regularly discovered in almost every software application. Even a moderately-sized network can have many different attack paths which an attacker could exploit to gain unauthorized network access. Amid this growing complexity, it is increasingly difficult for a human to fully and accurately understand the state of network security.

Currently, the evaluation and mitigation of security risks in an enterprise network is more an art than a science. System administrators operate by instinct and experience, often without any way to evaluate the full ramifications of any changes. Without an objective measurement of risk, there is no straightforward and reliable method to answer fundamental questions, such as "Where is our network most vulnerable?", "If we change A, will our network be more or less secure as a result?", and "How secure is my system?". These questions are important to answer when investing precious resources to improve the security of an enterprise network. Often times improvement of security also comes with a cost at functionality or ease of use; thus it is important to understand how much reduction in overall security risk a proposed change can achieve. To answer these questions requires a quantitative model of security with clear measurements of risk and easy comparison of different network states.

Much work has already been done in analyzing network configuration data and identifying network vulnerabilities to construct aggregate attack graphs [2, 7, 8, 13, 14, 17, 18, 19, 26, 27, 28, 30, 31, 36, 37, 38, 40]. Attack graphs illustrate the cumulative effect of attack steps, showing how individual steps can potentially enable an attacker to gain privileges deep within the network. The limitation of attack graphs, however, is the assumption that a vulnerability that exists can be exploited. In reality, there may be a wide range of probabilities that different attack steps could be profitably exploited by an attacker, dependent on the skill of the attacker and the difficulty of the exploit. Attack graphs show what is possible without any indication of what is likely.

Recently, there has been significant progress in standardizing and developing metrics for individual vulnerabilities, such as the Common Vulnerability Scoring System (CVSS) [24, 35]. This is known as a *component metric*, essentially a risk measurement indicating the likelihood that a vulnerability can be successfully exploited when all necessary preconditions are met. These risk measurements consider both specific qualities of vulnerabilities, such as the skill necessary to exploit the weakness, and known

[*]This work is supported by the U.S. National Science Foundation under Grant No. CNS-0716665. Any opinions, findings and conclusions or recomendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

information about the availability of exploit. This information can be used to assign a probability that the vulnerability will be exploited successfully when exposed to an attacker. The key limitation of CVSS is that the computed scores represent only the likelihood of success for individual attack steps, without consideration of the probability of actions performed to attain the preconditions necessary for an attack step. It is easy to see in an attack graph how an attacker might perform multi-step attacks to penetrate deeper into a network than may seem immediately possible. The probability of success for a multi-step attack is actually an aggregate calculation over the probabilities for each individual step in the path. Because each score reflects only the probability of a single step, CVSS (or any comparable component metric system) is insufficient in itself to fully quantify risk in an enterprise network. For example, a vulnerability may have a high CVSS score (indicating it represents high risk to a system when the vulnerability is exposed to an attacker). But the vulnerability may reside at a location that is highly difficult for an attacker to access. Likewise, a vulnerability may have a lower CVSS score but reside at a location that is relatively easy for an attacker to approach. To accurately measure security risk in an enterprise environment, both measurement at individual vulnerabilities' properties and their interplay must be taken into account. Since attack graphs represent logical inter-relationship among multiple attack steps, it is natural to combine attach graph and individual vulnerability metrics like CVSS to achieve comprehensive measurement of risk in an enterprise network.

There have been some attempts at measuring network security risk by combining attack graphs and individual vulnerability metrics. Frigault *et al.* [9, 10] propose converting attack graphs and the individual metrics into Bayesian Networks (BN) for computing the cumulative probability. While a BN can correctly account for shared dependencies in an attack graph, the major limitation of using BN is that it does not allow cycles which are common in attack graphs. While it is possible to "unfold" the attack graph into an acyclic representation, that approach is impractical because the dramatic increase in the size of graph structure will likely make BN reasoning inefficient. For example, an attack graph for a realistic network model with nine hosts contains 86 non-configuration nodes with 180 arcs; the unfolded attack graph for that same network contains 1,720 non-configuration nodes with 2598 arcs. Wang, *et al.* [41] recognize the presence of cycles in an attack graph and present useful ideas about propagating probability over cycles. However, their probability calculation seems to assume that probabilities along multiple paths leading to a node are independent, which does not hold in attack graphs.

In this paper, we propose a sound and practical approach to quantitatively measuring risk within an enterprise network, utilizing the attack graph structure in conjunction with the component metric input to calculate cumulative risk metrics indicating probability that a specific network privilege would be obtained by an attacker. Our major contribution is a *sound* model for computing cumulative risk metrics on attack graphs. The model has a well-defined semantics and a practical algorithm for computing the metrics.

One common criticism of security metrics arises from the fact that the input to the metric model (component metrics in this case) is often imprecise. And since there is no way to make the input numbers "correct", whatever the model can compute is meaningless. We disagree. While it is true that the component metrics, and as a result the computed cumulative metrics, are inevitably imprecise, that does not mean the result does not provide any useful information on security. Even if we can only use the numbers in the comparative nature that is already a big step forward. One may not care about the difference between 30% and 35%, but probably will notice a difference when seeing 30% and 90%. We contend that it is necessary to make initial steps toward quantitative risk assessment, rather than remaining constrained to precisely measurable properties such as counting the number of vulnerabilities. Those precisely measurable metrics may be "correct", but they are not useful since they do not reflect important properties people take into consideration when making security decisions. For example, knowing that a "high-risk" vulnerability is more likely to be successfully exploited than a "medium-" or "low-risk" vulnerability is important when deciding upon which one to patch first. Even though these valuations are at best rough estimates, that does not mean that they should not be used in practice. Work continues to progress in refining metrics for better capturing the properties of network vulnerabilities. A sound model for combining these component metrics can actually help make them more useful and even more precise: a cumulative assessment that is thought to be faulty can be traced back to the imprecision in the input component metrics, providing a feedback loop for refinement and calibration.

Developing a sound metric model based on attack graphs is not simple. Enterprise networks are typically laid out in such a way as to include a great deal of interconnectedness between network hosts. Thus, multiple attack paths leading to a given network privilege are rarely independent but more likely will share some dependencies. Furthermore, this interconnectedness will often lead to the appearance of cycles in an attack graph, but an accurate assessment of the probability that some privilege could be gained by an attacker should obviously not include a theoretical attack path in which an attacker can reach that privilege after having already gained it. Assuming monotonicity in the acquisition of network privileges [2], this attack path should be excluded from the calculations. We propose to treat both cyclic and acyclic attack graphs with proper management of shared dependencies and with a provably sound semantics. More specifically, assuming the input component metrics are correct, our model will produce an accurate assessment of the probability that an attacker can succeed in achieving a privilege.

One may wonder why a sound model for combining component metrics is so important given that the input is inevitably imprecise. Could an unsound model still provide useful result that is within the error bound inherent in the input? We think that an approximation algorithm with *provable error bounds* would be useful if that can turn into better performance and scalability. But we have not seen any development of such a model in the literature. This paper presents a first step in that direction, with the development of a *provably sound* model and algorithm for computing security risk in enterprise networks. Research on approximation algorithms will be explored in future work.

In Section 2, we will broadly describe the issues faced in quantitative risk assessment and, in Section 3, show how our approach handles these issues. We present some experimental results in Section 4 and address scalability concerns. We will review related works in Section 5 and conclude in Section 6 with a discussion of future work.

2 Problem Overview

An accurate assessment of security risk within an enterprise network must consider the structure and interconnectedness of the network. Much work has already been done in attack graphs to identify vulnerabilities within a network and to show how these vulnerabilities may be sequentially exploited to enable an attacker to gain privileges deep within the network. We can utilize attack graphs as a structural basis for risk assessment. We use the MulVAL attack graph toolkit [12, 27] in our work but our approach should be easily transferrable to other tools that produce attack graphs with similar semantics [13, 14, 31].

A full attack graph G will include three types of nodes: (1) attack-step nodes (collectively, set G_C), represented within the graph as logical AND-nodes. Each node in this set represents a single attack step which cannot be carried out unless all the predecessors (preconditions to the attack which are either configuration settings or network privileges) are satisfied; (2) privilege nodes (collectively, set G_D), represented within the graph as logical OR-nodes. Each node in this set represents a single network privilege which can be achieved when at least one of their predecessors is achieved. The predecessors represent various ways to obtain the privilege; (3) configuration nodes, represented within the graph as source nodes which have no predecessors. Each node in this set represents a fact about the current network configuration which is known to be true. For the purpose of risk assessment the configuration nodes can be removed since they have no variance in probability, leaving us with an AND/OR graph. In such an AND/OR graph an attack-step node is always preceded by privilege nodes and always has exactly one successor privilege node. A privilege node, however, may have multiple successor attack-step nodes, representing different options for further attacking the system.

A new privilege node will be added to serve as the root node in the graph; the new root of the graph, $G_R \in G_D$, represents no network privilege and there is an edge from G_R to all the attack-step nodes that do not require any prior privilege, *i.e.*, the nodes in G_C that has no predecessors after the configuration nodes are removed. In the subsequent technical discussion we assume G_R is the attacker's starting point, but the model will work for any node as assumed starting point. The set of all nodes with multiple successor nodes (must be privilege nodes), hereafter referred to as "branch nodes", is $G_B \subseteq G_D$.

Two examples of attack graphs are shown in Figures 1 and 2. Circular-shaped nodes are attack-step nodes and diamond-shaped nodes are privilege nodes. In both examples $G_R = P_0$. P_1 is a branch node in both graphs.

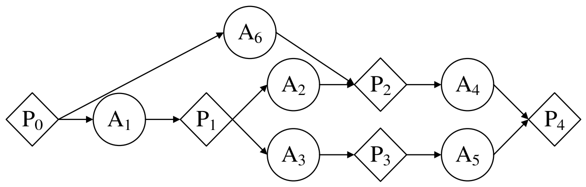

Figure 1: An attack graph without a cycle

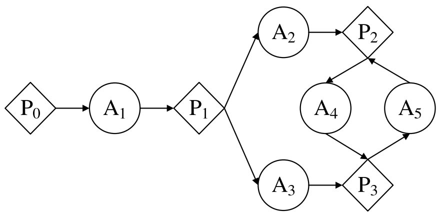

Figure 2: An attack graph with a cycle

A component metric is associated with each attack-step node. The metric represents the conditional probability that the single attack step will succeed when all of the prerequisite conditions are met. For example, the component metric for A_2 represents the probability that the attack step can succeed when an attacker already obtained privilege P_1. We use G_M to denote the set of relevant component metric values (*e.g.*, based on a CVSS score); each attack-step node $c \in G_C$ will have an assigned metric value denoted as $G_M[c]$. Thus, for some attack-step node e with predecessor set P, the probability of e given set P is equal to the component metric value: $Pr[e|P] = G_M[c]$. Additionally, the attack graph will have an assumed prior risk value, G_V, representing the probability of an attack being attempted against the network. G_V is associated with the root node of the graph and we assume $G_V = 1$ in the subsequent discussion.

2.1 Handling shared dependency

In an attack graph, it is common to see multiple attack paths leading to a single network privilege. In Figure 1, for example, privilege P_4 can be obtained by an attacker using either of two attack steps - A_4 or A_5. Privilege P_4 will be unobtainable if an attacker cannot successfully carry out the exploit A_4 or A_5. If the paths to A_4 and A_5 are independent, where $Pr[A_4]$ and $Pr[A_5]$ are the probabilities that A_4 and A_5, respectively, can be successfully carried out, we can easily calculate the probability that an attacker might gain privilege P_4 to be: $Pr[P_4] = Pr[A_4] + Pr[A_5] - Pr[A_4] \cdot Pr[A_5]$.

However, it is incorrect to assume that A_4 and A_5 are independent. Looking at Figure 1, it is easily seen that attack step A_4 is potentially affected by privilege P_1 and attack step A_5 fully depends upon it. Because of this shared dependence on P_1, A_4 and A_5 are not independent and the above formula would skew the effect that privilege P_1 has on the final result. In other words, assuming that all attack paths in an attack graph are independent will lead to unsound result in risk assessment.

To correctly account for shared dependencies among attack paths, we will employ the notion of *d-separation* within a causal network (such as a Bayesian Network) [15]. The concept of d-separation can be utilized to establish conditional independence between node sets. The risk assessment problem on attack graphs is a more specific problem than generic Bayesian reasoning; therefore, the concept of d-separation is specialized here for our specific application.

Definition 1 *Within an attack graph, for the purposes of risk assessment, two distinct nodesets A and B are d-separated by an intermediate nodeset $V \subseteq G_B$ (distinct from A and B) if along every diverging path between A and B, there is some $v \in V$ such that v is the point of divergence.*

Because of the structure of the attack graph, only *shared dependencies* (points of divergence in shared paths) need to be considered in the construction of a d-separating set. For example, in Figure 1, there is a diverging path $A_2 \leftarrow P_1 \rightarrow A_3$ between nodes A_2 and A_3; the point of divergence, P_1, d-separates these two nodes. Even though A_2 and A_3 are not independent (they are both influenced by P_1), when P_1 is fixed they become conditionally independent.

In an attack graph with the configuration nodes removed, the divergence nodes must be the branch nodes, which are all privilege nodes. Then the d-separating set D for two nodesets must be a subset of G_B and the elements in D must "block" all diverging paths between the two node sets. That means to find out a set of nodes that d-separates two node sets, we only need to consider branch nodes along the paths to the nodes. Take A_4 and A_5 in Figure 1 as an example; we only need to consider the branch nodes along the paths to A_4 and A_5 which is $D = \{P_0, P_1\}$. We can see from the graph that D d-separates A_4 and A_5, and once the nodes in D are fixed, A_4 and A_5 become conditionally independent. This makes the calculation of joint distribution easy.

Theorem 1 $\forall D, N \subseteq G_N, Pr[N] = \sum_D Pr[N|D] \cdot Pr[D]$

Proof:

$$\begin{aligned}\sum_D Pr[N|D] \cdot Pr[D] &= \sum_D Pr[N, D] \\ &= Pr[N]\end{aligned}$$

By Bayes theorem, $Pr[N|D] \cdot Pr[D]$ produces the joint probability $Pr[N, D]$. Summing over all possible values of D will *marginalize* D from the joint distribution $Pr[N, D]$ and we get $Pr[N]$.

Theorem 2 *Let D, N be node sets such that D d-separates any pair of nodes in N. Then all nodes $n \in N$ are* conditionally independent *given D: $Pr[N|D] = \prod_{n \in N} Pr[n|D]$.*

This theorem follows directly from the property of d-separation and conditional independence.

Using the above two theorems, we find a way to calculate the joint probability $Pr[\overline{A_4}, \overline{A_5}]$, which is needed to calculate $Pr[P_4]$. By Theorem 1, we can sum over all possible values of d-separating set $\{P_0, P_1\}$ to solve $Pr[\overline{A}_4, \overline{A}_5]$, and, applying Theorem 2, we can decompose a joint conditional distribution to singleton conditional probabilities.

$$\begin{aligned}Pr[P_4] &= 1 - Pr[\overline{A_4}, \overline{A_5}] \\ &= 1 - \sum_{P_0, P_1} Pr[\overline{A}_4, \bar{A}_5 | P_0, P_1] \cdot Pr[P_0, P_1] \\ &= 1 - \sum_{P_0, P_1} Pr[\overline{A}_4 | P_0, P_1] \cdot Pr[\overline{A}_5 | P_0, P_1] \cdot \\ &\qquad\qquad Pr[P_0, P_1]\end{aligned}$$

The formation and use of calculations will be discussed in greater detail in later sections.

2.2 Handling cycles

In Figure 2, some graph nodes comprise a cycle - $\{P_2, A_4, P_3, A_5\}$. When evaluating the probability of a node within a cycle, such as $Pr[P_2]$, we must be careful that node P_2 does not affect its own probability of occurrence. According to the graph, it is possible that an attack can use attack step A_2 to obtain privilege P_2, then use attack step A_5 to obtain privilege P_3, and then use attack step A_4 to obtain privilege P_2 again. Although this path does technically exist within the graph, it is meaningless for risk assessment and should have no influence on the probability that privilege P_2 might be obtained by an attacker. We must be able to evaluate nodes within the cycle while eliminating any cyclic influences. It is possible to unfold any cyclic graph into an equivalent acyclic graph such that each node appears exactly once in any path. But this procedure is not necessary if we apply a data flow analysis to the cyclic nodes so that we can evaluate the same probabilities as on the unfolded graph but without actually unfolding it.

3 Risk Assessment Approach

3.1 Definitions

For describing our approach of sound risk assessment, it is convenient to define and employ the following notations.

Function 1 *For a node $n \in G_N$, $\phi(n)$ represents the absolute probability that node n is true (i.e., the probability that the* *privilege/attack step represented by node n can be successfully gained/ launched). Similarly, $\phi(\overline{n})$ represents the probability that node n is false. $\phi(n) + \phi(\overline{n}) = 1$. For a node set $N \subseteq G_N$, $\phi(N)$ represents the probability that each $n \in N$ will be true or false as specified in set N.*

For example, in Figure 1, $\phi(P_4)$ is the probability that node P_4 is true, while $\phi(\overline{P}_4)$ is the probability that P_4 is false. Since P_4 must be either true or false, these two values must have a sum of one.

$\phi(\overline{A}_4, \overline{A}_5)$ is the joint probability that nodes A_4 and A_5 are both false; since there are shared dependencies between these nodes, calculating the precise value of $\phi(\overline{A}_4, \overline{A}_5)$ would require computing the conditional probability given a d-separating set.

Function 2 *For a node $n \in G_N$ and node set $A \subseteq G_N$, $\psi(A, n)$ represents the conditional probability that node n is true if the values of all $a \in A$ are fixed as specified in set A. For a node set $N \subseteq G_N$, $\psi(A, N)$ represents the conditional probability that each $n \in N$ will be true or false as specified in set N if the values of all $a \in A$ are fixed as specified in set A.*

For example, in Figure 1, $\psi(\{P_0, P_1\}, P_2)$ is the probability that P_2 is true given that P_0 and P_1 are true. This eliminates any influence that A_1 has on node P_2. $\psi(\{P_0, P_1\}, P_2) = e_2 + e_6 - e_2e_6$. Another example: $\psi(\{P_0, \overline{P}_1\}, P_2)$ is the probability that P_2 is true given that P_0 is true and P_1 is false. $\psi(\{P_0, \overline{P}_1\}, P_2) = e_6$.

Function 3 *For a node $n \in G_N$, $\chi(n) = \{b \mid b \in G_B$ and b appears in at least one attack path leading to node $n\}$.*

$\chi(n)$ is the set of all branch nodes within the graph that appear in at least one path to n and so may affect the probability of n. For example, in Figure 1, $\chi(A_6) = \{P_0\}$ and $\chi(A_2) = \{P_0, P_1\}$.

Definition 2 *Within the attack graph, for any node n, a logical dominator is any node d such that n is true only if d is true. This relationship is denoted $d \Leftarrow n$.*

Function 4 *For a node $n \in G_N$, $\delta(n) = \{d \mid d \in G_B$ and $d \Leftarrow n\}$.*

$\delta(n)$ is the set of all branch nodes that logically dominate (appear in all attack paths to) n, so that $\forall d \in \delta(n),\ \psi(\overline{d}, n) = 0$. In other words, n is false if any $d \in \delta(n)$ is false. In Figure 1, node $P_1 \Leftarrow A_5$ since A_5 is true only when P_1 is true: all attack paths to A_5 must first accomplish P_1. But P_1 does not dominate A_4, since there is a path to A_4 through A_6 which does not require P_1. Clearly, the set of nodes that must affect n is a subset of the all the nodes that may affect n, so $\delta(n) \subseteq \chi(n)$.

Employing these notations, we will now consider how to calculate the probability values for every node within an attack graph.

3.2 Formal specification of risk assessment

The following propositions set forth recursively-defined equations for the calculation of the functions defined in the previous section. The root node of the attack graph, G_R, will be initialized at the beginning of the algorithm and will serve as an anchor for the ϕ recursion. When calculating $\psi(A, n)$, the recursion must reach a point where $n \in A$ (so that n must be true), $\overline{n} \in A$ (so that n must be false), or $A \cap \chi(n) = \emptyset$ (so that n is conditionally independent of A); these base cases will serve to halt the recursion. The validity of the propositions and the termination of the recursive calculations has been formally proved but is not included in the paper due to space limitation.

Proposition 1 *For any privilege node $n \in G_D$ with immediate predecessor set W,*

$$\begin{aligned}\phi(n) &= 1 - \phi(\overline{W})\\ \psi(A, n) &= 1 - \psi(A, \overline{W})\\ \chi(n) &= \bigcup_{w \in W} \chi(w)\\ \delta(n) &= \bigcap_{w \in W} \delta(w)\end{aligned}$$

A privilege node n will be true when at least one of its predecessors are true; conversely, it will be false only when all of its predecessors are false. The χ set for n is the set of branch nodes that affect at least one path to some $w \in W$ and so at least one path to n; the δ set for n is the set of branch nodes that affect *all* paths to n (and so logically dominate n). Since the predecessors of a privilege node are all attack nodes, they cannot be branch nodes themselves.

Proposition 2 *For any privilege node $n \in G_D$ with immediate predecessor set $\{w\}$,*

$$\begin{aligned}\phi(n) &= \phi(w)\\ \psi(A, n) &= \psi(A, w)\\ \chi(n) &= \chi(w)\\ \delta(n) &= \delta(w)\end{aligned}$$

Proposition 2 is obviously a special case of Proposition 1, where the computation of values is simplified upon the assumption of exactly one predecessor, w. By Proposition 1, $\phi(n) = 1 - \phi(\overline{w})$, but since $\phi(w) + \phi(\overline{w}) = 1$, we know $\phi(n) = 1 - (1 - \phi(w) = \phi(w)$. $\chi(n) = \bigcup_{w \in W} \chi(w) = \chi(w)$. Since all paths to n lead through w, $\delta(n) = \delta(w)$.

Proposition 3 *For any attack-step node $n \in G_C$ with immediate* *predecessor set W*,

$$\begin{aligned}\phi(n) &= G_M[n] \cdot \phi(W) \\ \psi(A,n) &= G_M[n] \cdot \psi(A,W) \\ \chi(n) &= (\bigcup_{w \in W} \chi(w)) \cup (G_B \cap W) \\ \delta(n) &= (\bigcup_{w \in P} \delta(w)) \cup (G_B \cap W)\end{aligned}$$

When all predecessors are true, an attack step node n is true with conditional probability $G_M[n]$. Similarly, given set A, n is true with conditional probability $G_M[n]$ when all predecessors are conditionally true. The χ set for n is the set of branch nodes that affect at least one of its predecessors together with any of its predecessors that are branch nodes themselves. Since n requires that all predecessors be true, the δ set for n is the set of branch nodes that logically dominate *any* predecessor (and so logically dominate n) as well as *any* one of its predecessors that are branch nodes.

Proposition 4 *For any attack-step node $n \in G_C$ with immediate predecessor set $\{w\}$,*

$$\begin{aligned}\phi(n) &= G_M[n] \cdot \phi(w) \\ \psi(A,n) &= G_M[n] \cdot \psi(A,w) \\ \chi(n) &= \chi(w) \cup (G_B \cap \{w\}) \\ \delta(n) &= \delta(w) \cup (G_B \cap \{w\})\end{aligned}$$

Proposition 4 is obviously a special case of Proposition 3, where the computation of values is simplified upon the assumption of exactly one predecessor, w.

3.3 Example - Figure 1

We will now work through the sample graph shown in Figure 1, demonstrating our approach and showing its effectiveness at recognizing and correctly handling shared dependencies within the graph. This graph is acyclic, so we can recursively calculate the probability value for each node, utilizing previously calculated individual probability values and computing joint probabilities only as needed. To simplify the presentation, we use e_i to denote the component metric value for each attack-step node A_i, *i.e.* $\forall A_i \in G_C, G_M[A_i] = e_i$. A table showing all calculated values will be included at the end of this example.

We begin with the root node, P_0. The probability that node P_0 is true is G_V which is assumed to be 1. Thus $\phi(P_0) = 1$. As the root node, P_0 has no preceding nodes, so $\chi(P_0) = \{\ \}$ and $\delta(P_0) = \{\ \}$.

Now that we have calculated $\phi(P_0)$, we can calculate for either A_1 or A_6. Let us next calculate for A_6. Proposition 4 specifies the calculation for an attack-step node with exactly one predecessor. $\phi(A_6) = G_M[A_6] \cdot \phi(P_0) = e_6 \cdot 1 = e_6$; $\chi(A_6) = \delta(A_6) = \{P_0\}$.

We cannot yet evaluate node P_2, because not all of its predecessors have been evaluated. We will return, then, to evaluate node A_1. Similar to the calculation for A_6, $\phi(A_1) = G_M[A_1] \cdot \phi(P_0) = e_1$; $\chi(A_1) = \delta(A_1) = \{P_0\}$.

Table 1: Values calculated while solving for $\phi(\overline{A}_4, \overline{A}_5)$

N	$\phi(N)$
$\{P_0, P_1\}$	e_1
$\{P_0, \overline{P}_1\}$	$1 - e_1$
$\{\overline{P}_0, P_1\}$	0
$\{\overline{P}_0, \overline{P}_1\}$	0

D	$\psi(D, \overline{A}_4)$	$\psi(D, \overline{A}_5)$
$\{P_0, P_1\}$	$(1 - e_4(e_2 + e_6 - e_2e_6))$	$(1 - e_3e_5)$
$\{P_0, \overline{P}_1\}$	$(1 - e_4e_6)$	1
$\{\overline{P}_0, P_1\}$	1	1
$\{\overline{P}_0, \overline{P}_1\}$	1	1

Table 2: Risk assessment calculations for Figure 1

N	$\phi(N)$	$\delta(N)$	$\chi(N)$
P_0	1	$\{\}$	$\{\}$
P_1	e_1	$\{P_0\}$	$\{P_0\}$
P_2	$(e_1e_2 + e_6 - e_1e_2e_6)$	$\{P_0\}$	$\{P_0, P_1\}$
P_3	e_1e_3	$\{P_0, P_1\}$	$\{P_0, P_1\}$
P_4	$(e_4e_6 + e_1(e_2e_4 + e_3e_5) - e_1e_2e_4e_6 - e_1e_3e_4e_5(e_2 + e_6 - e_2e_6))$	$\{P_0\}$	$\{P_0, P_1\}$
A_1	e_1	$\{P_0\}$	$\{P_0\}$
A_2	e_1e_2	$\{P_0\}$	$\{P_0, P_1\}$
A_3	e_1e_3	$\{P_0, P_1\}$	$\{P_0, P_1\}$
A_4	$e_4(e_1e_2 + e_6 - e_1e_2e_6)$	$\{P_0\}$	$\{P_0, P_1\}$
A_5	$e_1e_3e_5$	$\{P_0, P_1\}$	$\{P_0, P_1\}$
A_6	e_1	$\{P_0\}$	$\{P_0\}$

We can now evaluate node P_1. Proposition 2 specifies the calculation for a privilege node with exactly one predecessor. Thus, $\phi(P_1) = \phi(A_1) = e_1$; $\chi(P_1) = \delta(P_1) = \{P_0\}$.

From this point, we could evaluate either A_2 or A_3. Let us next calculate for A_2. $\phi(A_2) = G_M[A_2] \cdot \phi(P_1) = e_2 \cdot e_1 = e_1e_2$; $\chi(A_2) = \delta(A_2) = \{P_0, P_1\}$.

Now both predecessors to P_2 have been solved and we can calculate for this node. Proposition 1 specifies the calculation for a privilege node with multiple predecessors. So, $\phi(P_2) = 1 - \phi(\{\overline{A}_2, \overline{A}_6\})$. In previous cases with single predecessors, we already knew the probability of the predecessor, but in this case we do not yet know the joint probability of $\phi(\{\overline{A}_2, \overline{A}_6\})$ and so must solve for it. To calculate $\phi(\{\overline{A}_2, \overline{A}_6\})$, we must find a d-separating set for these two nodes so that we can utilize Theorem 2 and 1. One such set can be found by taking the intersection of the χ sets for these nodes, so that $D = \chi(A_2) \cap \chi(A_6) = \{P_0\}$. D contains all branch nodes that diverge to paths leading to A_2 and A_6, which should be sufficient to d-separate the nodes (Definition 1). Using the set D, we can now solve for $\phi(\{\overline{A}_2, \overline{A}_6\})$:

$$\begin{aligned}
&\phi(\overline{A}_2, \overline{A}_6) \\
&= \sum_{P_0} \psi(\{P_0\}, \overline{A}_2, \overline{A}_6)\phi(\{P_0\}) && (Theorem\ 1) \\
&= \sum_{P_0} \psi(\{P_0\}, \overline{A}_2)\psi(\{P_0\}, \overline{A}_6)\phi(\{P_0\}) && (Theorem\ 2) \\
&= \psi(\{P_0\}, \overline{A}_2)\psi(\{P_0\}, \overline{A}_6)\phi(\{P_0\}) + \\
&\quad \psi(\{\overline{P}_0\}, \overline{A}_2)\psi(\{\overline{P}_0\}, \overline{A}_6)\phi(\{\overline{P}_0\}) \\
&= (1 - e_1e_2)(1 - e_6)(1) + (1)(1)(0) && (Propositions\ 1-4) \\
&= 1 - e_1e_2 - e_6 + e_1e_2e_6
\end{aligned}$$

Then, $\phi(P_2) = 1 - \phi(\{\overline{A}_2, \overline{A}_6\}) = 1 - (1 - e_1e_2 - e_6 + e_1e_2e_6) = e_1e_2 + e_6 - e_1e_2e_6$. Also, $\chi(P_2) = \chi(A_2) \cup \chi(A_6) = \{P_0, P_1\}$, and $\delta(P_2) = \delta(A_2) \cap \delta(A_6) = \{P_0\}$.

Nodes A_3, A_4, A_5, P_3 are calculated very similarly to nodes we've already seen here, so we will skip over the details of these. The resulting values are in Table 2.

Finally, we evaluate for node P_4, a privilege node with multiple predecessors, so again we will apply Proposition 1: $\phi(P_4) = 1 - \phi(\{\overline{A}_4, \overline{A}_5\})$. The d-separating set for $\{A_4, A_5\}$ is $D = \chi(A_4) \cap \chi(A_5) = \{P_0, P_1\}$. The term values for this calculation are shown in Table 1.

$$\begin{aligned}
\phi(\overline{A}_4, \overline{A}_5) = &\sum_{P_0, P_1} \psi(\{P_0, P_1\}, \overline{A}_4)\psi(\{P_0, P_1\}, \overline{A}_5)\phi(\{P_0, P_1\}) \\
&= \psi(\{P_0, P_1\}, \overline{A}_4)\psi(\{P_0, P_1\}, \overline{A}_5)\phi(\{P_0, P_1\}) + \\
&\quad \psi(\{P_0, \overline{P}_1\}, \overline{A}_4)\psi(\{P_0, \overline{P}_1\}, \overline{A}_5)\phi(\{P_0, \overline{P}_1\}) + \\
&\quad \psi(\{\overline{P}_0, P_1\}, \overline{A}_4)\psi(\{\overline{P}_0, P_1\}, \overline{A}_5)\phi(\{\overline{P}_0, P_1\}) + \\
&\quad \psi(\{\overline{P}_0, \overline{P}_1\}, \overline{A}_4)\psi(\{\overline{P}_0, \overline{P}_1\}, \overline{A}_5)\phi(\{\overline{P}_0, \overline{P}_1\}) \\
&= (1 - e_4(e_2 + e_6 - e_2e_6))(1 - e_3e_5)(e_1) + \\
&\quad (1 - e_4e_6)(1)(1 - e_1) + (1)(1)(0) + (1)(1)(0) \\
&= 1 - e_1e_2e_4 + e_1e_2e_4e_6 - \\
&\quad e_1e_3e_5 - e_4e_6 + e_1e_3e_4e_5(e_2 + e_6 - e_2e_6)
\end{aligned}$$

Then, $\phi(P_4) = 1 - \phi(\overline{A}_4, \overline{A}_5) = e_4e_6 + e_1(e_2e_4 + e_3e_5) - e_1e_2e_4e_6 - e_1e_3e_4e_5(e_2 + e_6 - e_2e_6)$. Also, $\chi(P_4) = \chi(A_4) \cup \chi(A_5) = \{P_0, P_1\}$, and $\delta(P_2) = \delta(A_2) \cap \delta(A_6) = \{P_0\}$.

We have now solved for the probability of each node in the graph, and the computed ϕ values for individual nodes are shown in Table 2, together with the χ and δ sets for each node. In our implementation, joint and conditional probability values are calculated only as needed, to reduce the amount of computation performed. We also apply dynamic programming techniques to cache the calculated values to avoid repeating the same computation.

3.4 Example - Figure 2

In the previous example, we showed that the probabilities for graph nodes depend on the probabilities of their predecessors, so a recursive approach can be employed for this calculation. Within a cycle, however, recursing backward through predecessor sets will create an infinite loop. It is clear that a different approach is needed for cyclic nodes.

Figure 2 contains an attack graph that we will now use as an aid to explain and demonstrate quantitative risk assessment over an attack graph containing cycles. This attack graph contains one cycle, node set $\{P_2, P_3, A_4, A_5\}$. Figure 3 shows an equivalent representation of the same attack graph. This unfolded attack graph is acyclic, containing all of the unique, acyclic paths that traverse the cycle. Node P_2, for example, can be reached as P_{2A} or P_{2B}; the dotted-line arcs indicate that reaching either of these instances means that P_2 has been reached. In other words, P_{2A} and P_{2B} can be viewed as "partial values" for P_2 and P_2 is true when either of them is true.

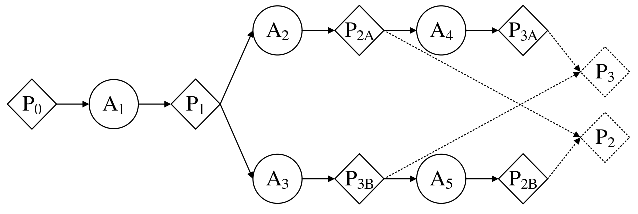

Figure 3: Example: Cyclic Attack Graph - Unfolded

We will now work through the sample graph shown in Figure 2, demonstrating our approach and showing its effectiveness at recognizing and correctly handling cycles within the graph. A table showing all calculated values will be included at the end of this example.

Intuitively, we will identify all acyclic paths traversing the cycle and use this knowledge to logically identify unique possible instances of cyclic nodes. We can then d-separate over the set of reaching paths to calculate the probability that the cyclic node will be reached.

Nodes P_0, A_1, P_1, A_2, A_3 will be calculated very much as demonstrated in Section 3.3 and so we will not go through the detailed calculations for those nodes. Once these have been calculated, however, the remaining graph nodes comprise a cycle and therefore must be handled differently. In our implementation, we have chosen to use Tarjan's algorithm for the identification of strongly connected graph components [39].

We can simplify the handling of cycles by calculating values only for cyclic nodes with multiple immediate predecessors [41]. Considering Propositions 2 and 4, it is easily seen that the probability for a node with exactly one predecessor is dependent only on the value of that predecessor, so that when the probability of that predecessor node is known, the recursion in the calculation of the probability for a single-predecessor node will stop at that predecessor. Thus, the potential for an infinite recursion through the cycle (as with a multi-predecessor node) is eliminated. In this example, the cyclic nodes with multiple predecessors are P_2 and P_3.

Table 3: Values calculated while solving for $\phi(\overline{P}_{2A}, \overline{P}_{2B})$

N	$\phi(N)$
$\{A_2, A_3\}$	$e_1e_2e_3$
$\{A_2, \overline{A}_3\}$	$e_1e_2(1-e_3)$
$\{\overline{A}_2, A_3\}$	$e_1e_3(1-e_2)$
$\{\overline{A}_2, \overline{A}_3\}$	$e_1(1-e_2)(1-e_3)$

D	$\psi(D, \overline{P}_{2A})$	$\psi(D, \overline{P}_{2B})$
$\{A_2, A_3\}$	0	$1-e_5$
$\{A_2, \overline{A}_3\}$	0	1
$\{\overline{A}_2, A_3\}$	$1-e_5$	$1-e_5$
$\{\overline{A}_2, \overline{A}_3\}$	1	1

Table 4: Risk assessment calculations for Figure 2

N	$\phi(N)$	$\delta(N)$	$\chi(N)$
P_0	1	{}	{}
P_1	e_1	{}	{}
P_2	$e_1e_2 + e_1e_3e_4 - e_1e_2e_3e_4$	−	−
P_3	$e_1e_3 + e_1e_2e_5 - e_1e_2e_3e_5$	−	−
A_1	e_1	{}	{}
A_2	e_1e_2	$\{P_1\}$	$\{P_1\}$
A_3	e_1e_3	$\{P_1\}$	$\{P_1\}$
A_4	$e_4(e_1e_3 + e_1e_2e_5 - e_1e_2e_3e_5)$	−	−
A_5	$e_5(e_1e_2 + e_1e_3e_4 - e_1e_2e_3e_4)$	−	−

First, we will trace all acyclic paths through the cycle, to determine all valid ways that these nodes can be reached. This trace essentially performs a logical unfolding of the graph, marking unique passes through each node. The acyclic paths through this cycle are:

$$P_{2A} = \{A_2\} \qquad P_{3A} = \{A_2, P_{2A}, A_4\}$$
$$P_{3B} = \{A_3\} \qquad P_{2B} = \{A_3, P_{3B}, A_5\}$$

There are two unique instances of node P_2 in this logical unfolding of the graph, P_{2A} and P_{2B}. The probability that node P_2 is true will equal the probability that at least one of these instances is true, or $\phi(P_2) = 1 - \phi(\overline{P}_{2A}, \overline{P}_{2B})$.

Definition 3 *For some set $C \subset G_N$ comprising a strongly connected component (cycle) within the graph, the* entry nodes *are the set of nodes Q such that $Q \cap C = \{\}$ and $\forall q \in Q, \exists c \in C, (q, c) \in G_E$. That is, the entry nodes are not in the cycle, but each entry node does have an arc leading into the cycle.*

To calculate $\phi(\overline{P}_{2A}, \overline{P}_{2B})$, we must calculate the joint probability of the set of entry nodes $\{A_2, A_5\}$ and we will also need to identify a d-separating set D within the cycle, to ensure that the instances of P_2 are conditionally independent. In this case, $D = Path(P_{2A}) \cap Path(P_{2B}) = \emptyset$; because the cycle is so small, these partial paths are already conditionally independent, given the entry points into the cycle. The formula of computation is shown below and the values calculated for each term in the equations are shown in Table 3.

$$\begin{aligned}
&\phi(\overline{P}_{2A}, \overline{P}_{2B}) \\
= &\sum_{A_2, A_3} \phi(\{A_2, A_3\})\psi(\{A_2, A_3\}, \overline{P}_{2A})\psi(\{A_2, A_3\}, \overline{P}_{2B}) \\
= &\ \phi(\{A_2, A_3\})\psi(\{A_2, A_3\}, \overline{P}_{2A})\psi(\{A_2, A_3\}, \overline{P}_{2B}) + \\
&\ \phi(\{A_2, \overline{A}_3\})\psi(\{A_2, \overline{A}_3\}, \overline{P}_{2A})\psi(\{A_2, \overline{A}_3\}, \overline{P}_{2B}) + \\
&\ \phi(\{\overline{A}_2, A_3\})\psi(\{\overline{A}_2, A_3\}, \overline{P}_{2A})\psi(\{\overline{A}_2, A_3\}, \overline{P}_{2B}) + \\
&\ \phi(\{\overline{A}_2, \overline{A}_3\})\psi(\{\overline{A}_2, \overline{A}_3\}, \overline{P}_{2A})\psi(\{\overline{A}_2, \overline{A}_3\}, \overline{P}_{2B}) \\
= &\ (0) + (0) + (e_1e_3(1-e_2))(1)(1-e_4) + \\
&\ (e_1(1-e_2)(1-e_3) + 1 - e_1)(1)(1) \\
= &\ 1 - e_1e_2 - e_1e_3e_4 + e_1e_2e_3e_4
\end{aligned}$$

So, $\phi(P_2) = 1 - (1 - e_1e_2 - e_1e_3e_4 + e_1e_2e_3e_4) = e_1e_2 + e_1e_3e_4 - e_1e_2e_3e_4$. By a similar calculation, $\phi(P_3) = e_1e_3 + e_1e_2e_5 - e_1e_2e_3e_5$. Once these have been solved, it is easy to see that $\phi(A_4) = G_M[A_4] \cdot \phi(P_3) = e_4(e_1e_3 + e_1e_2e_5 - e_1e_2e_3e_5)$ and $\phi(A_5) = G_M[A_5] \cdot \phi(P_2) = e_5(e_1e_2 + e_1e_3e_4 - e_1e_2e_3e_4)$.

The full results are shown in Table 4. Nodes P_2, P_3, A_4, A_5 do not have χ or δ sets, because these values are not used for evaluation within a cycle.

3.5 Algorithms

Although space constraints prevent us from including the full algorithms for the implementation and the formal proofs showing the correctness of our approach, we stress that this approach is sound and will always return accurate probabilities, based on the network model and the component metrics given as input. In the Appendix we show part of the pseudocode of our algorithms that correspond to the four propositions described earlier in the section. Below we give an informal description of the whole algorithm.

Our algorithm first initializes the root node ($\phi(G_R) = G_V$) and then identifies all graph cycles. The algorithm iterates over the set of nodes in the attack graph, seeking in each iteration a node or set of nodes ready for evaluation. A single node n (not contained within a cycle) is ready for evaluation when all of its immediate predecessors W have been evaluated. A cyclic set C is ready for evaluation when all of its entry nodes Q have been evaluated.

If a node n is ready to be evaluated, then its probability is calculated using its predecessor set W, employing the principles set forth in Propositions 1-4. If n is a single-predecessor node ($W = \{w\}$), then the probability of n is based upon $\phi(w)$ and, if $n \in G_C$, $G_M[n]$. If n is a multiple-predecessor node ($W = \{w_0, \ldots, w_n\}$), then a d-separating set D is identified for all $w \in W$. If $n \in G_D$, then $\phi(\mathrm{n}) = 1 - \phi(\overline{W})$; the joint probability $\phi(\overline{W})$ is calculated by marginalizing the effect of D over W. If $n \in G_C$, then $\phi(\mathrm{n}) = G_M[n] \cdot \phi(W)$; the joint probability $\phi(W)$ is also calculated using d-separating set D.

If a cyclic subset C is ready to be evaluated, then the algorithm calculates the probability for each of the nodes within the cycle. First, all unique, acyclic paths from the entry nodes to nodes in C are traced and stored as sets. Next, each multi-predecessor node $m \in C$ is evaluated, using the different enabling paths through C to identify unique instances of m and d-separating over these paths, so that node m is reachable whenever at least one instance of m is reachable. Finally, each single-predecessor $s \in C$ is evaluated using Propositions 2 and 4, by which the probability of s relies on the (already calculated) probability of its sole predecessor.

Once a cycle has been evaluated, the set of cyclic nodes can be removed from the attack graph and replaced with a virtual node representing that set. This node can be added to the χ and δ sets for successor nodes to maintain an acyclic graph, but the original set of nodes can be referenced when necessary, such as when calculating the likelihood for a node with multiple predecessors in the cycle.

It can be proven that in each iteration, some node n or cyclic set C is ready for evaluation. When all nodes have been evaluated, the algorithm terminates. To reduce computational overhead, conditional and joint probabilities are calculated only as needed and all calculated values are stored for possible later reference.

4 Implementation and Testing

A preliminary implementation of our algorithms, written in the Python language, has been the basis for our testing to date. Figure 4 shows a sample enterprise network, which we will use to demonstrate the application of our risk assessment algorithms.

Hosts and Accessibility: There are five hosts (or groups of hosts) with this enterprise network. The DMZ subnet and the VPN server are directly accessible from the Internet. The DMZ subnet, which contains only the web server, is permitted to access

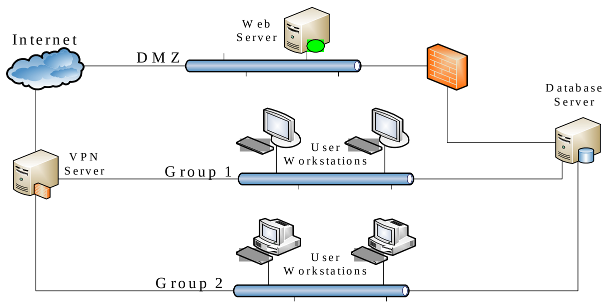

Figure 4: Example enterprise network

the database server through the firewall. The VPN server, which also serves as a firewall to protect internal subnets, is able to access the two subnets of user workstations, Group 1 and Group 2. Although there are multiple machines in each of these subnets, we model each group as a single host since the hosts have identical configurations. Groups 1 and 2 also have permissions to access the database server, each through a unique application and port. The database server is permitted to access the user workstation subnets.

Vulnerabilities: All of the machines in this network have at least one vulnerability that potentially allows for remote exploitation, given an attacker code-execution privileges on the machine. The database server actually hosts three vulnerabilities, one on each of the active applications accessed by the web server, Group 1, and Group 2. Any one of these faults are sufficient for an attacker to gain control of the database server.

We will assume that an attacker's goal is to gain code-execution privileges on the database server. The full MulVAL-generated attack graph for this enterprise network model is shown in Figure 5. The attack graph includes twelve privilege nodes, fourteen attack-step nodes, and twenty-two configuration nodes.

For simplicity, we will assume that every vulnerability in the network has a component metric of 0.75. In other words, for each vulnerability in the network, when all necessary preconditions are met, an attacker will succeed in an exploit utilizing that vulnerability with 75% likelihood.

Considering the network topology, a network administrator would likely assume that the web server and VPN server at the most vulnerable, while the groups of user workstations and the database server are more secure, since these are not directly accessible from outside the network. This assumption seems intuitively correct, but a careful and complete evaluation of the probability of exploitation for each machine shows a different state of affairs.

The full risk assessment results for Figure 4 are shown in the first data column in Table 5. Here we see that the database server is the *most* vulnerable machine in the network, due to the several distinct attack paths leading to an attacker gaining privilege on this machine. Because the database server can be compro-

Table 5: Risk assessment calculations for Figure 4 (columns reflect different scenarios)

Host	Probability of compromise (all component metrics equal)	Probability of compromise (VPN metric lower)	Probability of compromise (patch vulnerability)
Web server	0.75	0.75	0.75
VPN server	0.75	0.25	0.75
User Group 1	0.5625	0.1875	0.5625
User Group 2	0.5625	0.1875	0.5625
Database server	0.8278	0.6509	0.6064

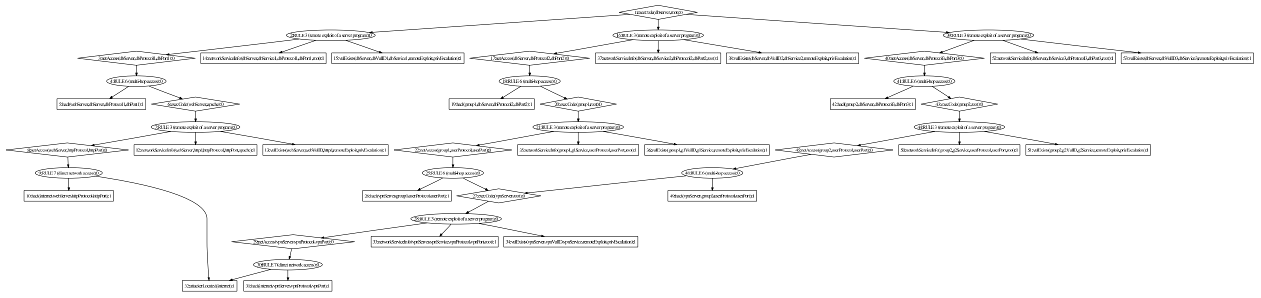
Figure 5: Attack graph for the example

mised by a successful exploitation of one or more of these possible attack paths, the chance that an attacker will successfully gain code-execution privileges is higher than might be intuitively expected.

The risk assessment values are for the given component metrics and the current network configuration. Alterations in either of these inputs can change the risk picture within the network. Consider the same scenario, except that the probability of success for the vulnerability on the VPN server is 0.25, much lower than the other probabilities, each 0.75. The change in this component metric value will affect not only the risk assessment for the VPN server itself, but also for all hosts that have attack paths through the VPN server.

The second column in Table 5 shows the risk assessment values with the updated VPN metric value. As can be seen in these results, the lower risk value for the VPN server affects several attack paths leading to different nodes. The probability that the web server will be exploited remains unchanged, since no attack path leads from the VPN server to the web server. The other probability values have changed; the probability values are reduced for the VPN server, both user groups, and even the database server.

Network configuration changes can also have drastic effects on the security risk. For example, suppose that the software vulnerability in the database service accessed from the web server is patched, so that there is now no attack path leading from the web server to the database server. Employing again the universal 0.75 component metric value for all the rest of the vulnerabilities, the risk assessment results for this scenario is shown in the third column in Table 5. As seen in these results, the removal of one of the three attack steps leading to privileges on the database server has noticeably reduced the probability that an attacker might compromise the database server. This will be useful information when evaluating which vulnerability to patch first to gain the maximum reduction in security risk.

4.1 Scalability

In order to test the scalability of our approach, we constructed several testing models based on networks of varying sizes and complexity, created MulVAL input files representing each network, and evaluated them with a preliminary implementation of our algorithms. The results of these tests are displayed in Table 4.

Models A, B, and C contain realistic network models. Model C is based on a real control system network for power grids, which cannot be revealed here. Model B is adapted from model C, adding an additional host and altering the accessibility configurations between hosts. Model A is still more sparsely connected, so that no large cycles predominate in the graph.

Models D, E, and F contain cliques, in which every network host is accessible from every other host, so a single strongly-connected component (cycle) contains almost every node in the attack graph (except the configuration nodes and the privilege representing an attacker's initial access to the network). In these scenarios, each host has exactly one remote-exploitable vulnerability that can provide an attacker with code-execution privileges.

Table 6: Scalability test results - risk assessment

Scenario type	Testing model	Number of network hosts	Number of graph nodes in largest cycle	Multi-predecessor cycle node evaluation time (average)	Total run-time
Realistic models	A	9	28	$<$ 0:01	0:04
	B	10	40	0:02	0:15
	C	9	46	0:46	8:31
Cliques	D	5	40	0:01	0:06
	E	7	54	0:25	2:33
	F	8	70	7:45	54:40

These models represent the worst-case situation in network connectivity, because a more strongly-connected attack graph will include a greater number of acyclic paths reaching to each graph node, making the assessment more complex. In a clique, each additional node will dramatically increase the number of acyclic paths per node; the number of entry nodes and the size of the d-separating set also increase.

The limiting factors in the current algorithm and implementation are the size of the d-separating set (the number of nodes which must be marginalized in calculating conditional probability values) and the number of paths that must be considered in the calculation of each multi-predecessor node within a cycle. As any of these increase, the number of recursive calls made by the algorithm increases and so the evaluation time also grows correspondingly, as seen in the table.

5 Related Works

The issue of security metrics has long attracted much attention [16, 21, 22], and recent years have seen significant effort on the development of quantitative security metrics [1, 4, 6, 20, 25, 29, 32]. There is also skepticism on the feasibility of security metrics given the current software and system architectures [5]. While we may still be far from achieving an objective quantitative metric for a system's overall security, a practical method for quantifying risks in an enterprise network based on known information about potential vulnerabilities is highly valuable in practice. Albeit only one aspect of a system's overall security, such risk metrics can produce highly needed automated guidance on how to spend the limited IT management resources and how to balance security and usability in a meaningful manner. Our work provides the important enabling technology for automated decision making through sound models and algorithms for quantifying security risks using attack graphs and component metrics such as CVSS.

Attack graphs have emerged as a main-stream techinque for enterprise network vulnerability analysis [2, 7, 8, 13, 14, 17, 18, 19, 26, 27, 28, 30, 31, 36, 37, 38, 40]. Recent years have also seen effort on computing various metrics from attack graphs [10, 34, 41, 42, 43, 44]. Our work built upon results from some of these previous works and is unique in that it provides a sound model and algorithm that accurately accounts for both cyclic and shared dependencies in attack graphs.

The Common Vulnerability Scoring System (CVSS) [24, 35] provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. CVSS consists of three metric groups: Base, Temporal and Environmental. Each of these groups produces a vector of compressed textual representation that reflects various properties of the vulnerability (*metric vector*). A formula takes the vector and produces a numeric score in the range of 0 to 10 indicating the severity of the vulnerability. The most important contribution of CVSS is the metric vector: it provides a wide range of vulnerability properties that are the basis for deriving the numerical scores. The main limitation of CVSS is that it provides scoring of individual vulnerabilities but it does not provide a methodology on how to aggregate the metrics for a set of vulnerabilities in a network to provide an overall network security score. The overall security of a network configuration running multiple services cannot be determined by simply counting the number of vulnerabilities or adding up the CVSS scores. Our work provides a sound mathematical model based on attack graphs that can be used to aggregate CVSS metrics to reflect the cumulative effect of vulnerabilities in an enterprise environment.

Frigault *et al.* [9, 10] utilizes the combination of attack graphs and Bayesian Networks (BN) in measuring network security. The major limitation of using BN is that it does not allow cycles which are common in attack graphs. Our approach does not use BN reasoning as a black box. Instead, we utilize the key concept of d-separation in BN inference and design customized algorithms for probabilistic reasoning on attack graphs. Such an approach not only can handle cycles within the context of risk assessment, but also takes into consideration special properties of risk assessment (*e.g.* that no real-time evidence needs to be considered, the monotonicity property, and so on) which can eliminate some unnecessary overhead and complexity in a standard BN inference engine.

Wang, *et al.* [41] recognize the presence of cycles in an attack graph and present useful ideas about propagating probability over cycles. However, their probability calculation seems to assume that probabilities along multiple paths leading to a node are independent, which is not generally true for attack graphs. Our approach correctly handles both cycles and shared dependencies in attack graphs.

Anwar, *et al.* [3] introduce an approach to quantitatively assessing security risks for critical infrastructures. The approach captures both static and dynamic properties in the network, contained in network and workflow models. However, the work did not provide a mathematical model to explain what the calculated metrics mean. Our risk metric has a clear semantics which is the likelihood an attacker can succeed in achieving a privilege or carrying out an attack. Our metric model provides a sound linkage between the input component metrics and the output cumulative metrics.

Sawilla and Ou design an extended Google PageRank algorithm and apply it to attack graphs to calculate numeric ranks for the graph nodes in terms of their importance for an attacker to achieve his perceived goals [34]. Mehta *et al.* also apply Google PageRank on a different type of attack graphs [23]. Numeric values computed from PageRank-like algorithms only indicate *relative* ranks and cannot be used to quantify absolute security risks [33]. Our work provides a sound and practical method for quantifying the absolute security risks in an enterprise network.

Wang, *et al.* [8] introduce an approach that assumes cost metrics are present for all nodes in an attack graph and uses this information to identify a minimum-cost network hardening solution. Dewri, *et al.* [8] formulate security hardening as a multi-objective optimization problem, using a genetic algorithm to search for an optimal solution based on costs of security hardening and potential damage. Homer and Ou [11] demonstrate the effectiveness of using MinCostSAT as a basis for automated network reconfiguration, with numeric cost values being assigned to each configuration setting and reachable privilege in the attack graph. The risk metrics developed from our work can be used to derive numeric weight inputs to these algorithms for optimal security hardening.

6 Conclusion and Future Work

We have presented an approach to quantifying security risk in an enterprise network, using attack graphs and component metrics such as CVSS. Our approach is sound in that given component metrics which characterizes the success likelihood of individual vulnerabilities, the model computes a numeric value representing the cumulative likelihood for an attacker to succeed in gaining a privilege or carrying out an attack in the specific network. Our method handles both cyclic and shared dependencies in attack graphs correctly, surpassing previous efforts on this problem. Preliminary experimental results show the effectiveness and practicality of the approach.

For future work, we believe that opportunity exists for further refinement of the algorithm toward lower run-time and more efficient handling of cycles and larger data sets. The running time of the algorithm depends less on the size of the data set than on its interconnectedness. A large but weakly connected data set will probably be calculated faster than a smaller but more strongly connected data set. We will study abstraction techniques for better handling larger and more strongly connected network models. In the implementation, there is also an opportunity to exploit parallelism in the algorithm to utilize multiple CPU cores or even cluster computing.

We plan to create a more robust implementation of this algorithm and to continue experimentation using real-world network data, to better determine the effectiveness of this approach in realistic situations. Another concern in the metric system is the reliability of the input component metrics. We believe that the inherent soundness of our approach to calculating risk in the network will serve to highlight gross errors in the input data. We plan to study how to use our metric model to identify imprecision in the input component metrics, and methodologies for calibrating the component metrics.

References

[1] E. Al-Shaer, L. Khan, and M. S. Ahmed. A comprehensive objective network security metric framework for proactive security configuration. In *ACM Cyber Security and Information Intelligence Research Workshop*, 2008.

[2] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, Washington, DC, November 2002.

[3] Z. Anwar, R. Shankesi, and R. H. Campbell. Automatic security assessment of critical cyber-infrastructures. In *Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, July 2008.

[4] 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.

[5] S. Bellovin. On the brittleness of software and the infeasibility of security metrics. *IEEE Security & Privacy*, 2006.

[6] E. Chew, M. Swanson, K. Stine, N. Bartol, A. Brown, and W. Robinson. *Performance Measurement Guide for Information Security*. National Institute of Standards and Technology, July 2008. NIST Special Publication 800-55 Revision 1.

[7] J. Dawkins and J. Hale. A systematic approach to multi-stage network attack analysis. In *Proceedings of Second IEEE International Information Assurance Workshop*, pages 48 – 56, April 2004.

[8] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *14th ACM Conference on Computer and Communications Security (CCS)*, 2007.

[9] M. Frigault and L. Wang. Measuring network security using Bayesian network-based attack graphs. In *Proceedings of the 3rd IEEE International Workshop on Security, Trust, and Privacy for Software Applications (STPSA'08)*, 2008.

[10] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic Bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, 2008.

[11] J. Homer and X. Ou. SAT-solving approaches to context-aware enterprise network security management. *IEEE JSAC Special Issue on Network Infrastructure Configuration*, 2009.

[12] J. Homer, A. Varikuti, X. Ou, and M. A. McQueen. Improving attack graph visualization through data reduction and attack grouping. In *The 5th International Workshop on Visualization for Cyber Security (VizSEC)*, 2008.

[13] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, December 2006.

[14] S. Jajodia, S. Noel, and B. O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challanges*, chapter 5. Kluwer Academic Publisher, 2003.

[15] F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Verlag, 2 edition, 2007.

[16] D. G. Jr., K. S. Hoo, and A. Jaquith. Information security: Why the future belongs to the quants. *IEEE SECURITY & PRIVACY*, 2003.

[17] W. Li, R. B. Vaughn, and Y. S. Dandass. An approach to model network exploitations using exploitation graphs. *SIMULATION*, 82(8):523–541, 2006.

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

[19] R. P. Lippmann, K. W. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham. Evaluating and strengthening enterprise network security using attack graphs. Technical Report ESC-TR-2005-064, MIT Lincoln Laboratory, October 2005.

[20] P. Manadhata, J. Wing, M. Flynn, and M. McQueen. Measuring the attack surfaces of two FTP daemons. In *Proceedings of the 2nd ACM workshop on Quality of protection*, 2006.

[21] J. McHugh. Quality of protection: measuring the unmeasurable? In *Proceedings of the 2nd ACM workshop on Quality of protection (QoP)*, Alexandria, Virginia, USA, 2006.

[22] J. McHugh and J. Tippett, editors. *Workshop on Information-Security-System Rating and Ranking (WISSRR)*. Applied Computer Security Associates, May 2001.

[23] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing. Ranking attack graphs. In *Proceedings of Recent Advances in Intrusion Detection (RAID)*, September 2006.

[24] P. Mell, K. Scarfone, and S. Romanosky. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. Forum of Incident Response and Security Teams (FIRST), June 2007.

[25] National Institute of Standards and Technology. *Technology assessment: Methods for measuring the level of computer security*, 1985. NIST Special Publication 500-133.

[26] X. Ou. *A logic-programming approach to network security analysis*. PhD thesis, Princeton University, 2005.

[27] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security (CCS)*, pages 336–345, 2006.

[28] X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium*, 2005.

[29] 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*, 2006.

[30] 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*, pages 71–79. ACM Press, 1998.

[31] D. Saha. Extending logical attack graphs for efficient vulnerability analysis. In *Proceedings of the 15th ACM conference on Computer and Communications Security (CCS)*, 2008.

[32] M. Salim, E. Al-Shaer, and L. Khan. A novel quantitative approach for measuring network security. In *INFOCOM 2008 Mini Conference*, 2008.

[33] R. Sawilla and X. Ou. Googling attack graphs. Technical report, Defence R & D Canada – Ottawa, 2007.

[34] R. Sawilla and X. Ou. Identifying critical attack assets in dependency attack graphs. In *13th European Symposium on Research in Computer Security (ESORICS)*, Malaga, Spain, October 2008.

[35] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright, and S. Romanosky. *CVSS: A Common Vulnerability Scoring System*. National Infrastructure Advisory Council (NIAC), 2004.

[36] O. Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Carnegie Mellon, April 2004.

[37] 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*, pages 254–265, 2002.

[38] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, volume 2, June 2001.

[39] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[40] T. Tidwell, R. Larson, K. Fitch, and J. Hale. Modeling Internet attacks. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, West Point, NY, June 2001.

[41] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In *Proceedings of The 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC'08)*, 2008.

[42] L. Wang, A. Singhal, and S. Jajodia. Measuring network security using attack graphs. In *Third Workshop on Quality of Protection (QoP)*, 2007.

[43] L. Wang, A. Singhal, and S. Jajodia. Measuring the overall security of network configurations using attack graphs. In *Proceedings of 21th IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC'07)*, 2007.

[44] L. Williams, R. Lippmann, and K. Ingols. Garnet: A graphical attack graph and reachability network evaluation tool. In *The 5th International Workshop on Visualization for Cyber Security (VizSEC)*, 2008.

A Appendix

Algorithm 1 presents the core algorithm for risk assessment over an attack graph. This algorithm consists primarily of a controlling loop that will iteratively consider each individual non-cyclic node or set of cyclic nodes in the graph. For individual nodes, the calculations detailed in Propositions 1-4 are applied; for cyclic node sets C, another algorithm $evalCycle$ is called to assess each $c \in C$. The algorithm will terminate when the assessment is complete.

Algorithm 2 calculates the joint probability of an acyclic node set N. If the value has previously been computed, the stored result is retrieved and returned, to prevent redundant calculations. Assuming that the needed value is not already known, the algorithm identifies a d-separating set D such that all $n \in N$ are conditionally independent given D (Theorem 2) and then marginalizes the influence of D on N (Theorem 1).

Algorithm 3 calculates the conditional probability of some node set N given d-separating node set D. Because D d-separates N, each $n \in N$ is conditionally independent from the remainder of the set and so can be solved individually. The algorithm checks several base cases to see if n is necessarily true, necessarily false, or independent of D; if none of these cases match, the algorithm is recursively called on the predecessor set of n.

Other algorithms, including $evalCycle$ and supporting algorithms for the the handling of cyclic node sets, are omitted here due to space constraints. Similarly, formal proofs of correctness for every algorithm have been constructed but cannot be included here.

Algorithm 2 Pseudocode for computing $evalProb(N)$

```
Require: Parameter $N$, such that
  $N = \{n_0, n_1, ..., n_j\} \subseteq G_N$, such that $\forall n_i \in N, \phi(n)$ has already been evaluated
  if $\phi(N)$ previously calculated then
      return $\phi(N)$ from table of stored values
  end if
  { Find d-separating set $D$ for node set $N$ }
  $D \leftarrow \bigcup_{m,n \in N} \chi(m) \cap \chi(n)$
  if $D = \emptyset$ then
      { No d-separating set, so nodes are independent }
      return $\prod_{n \in N} \phi(n)$
  else
      { Calculate conditional probabilities given $D$ }
      return $\sum_{D} evalCondProb(D, N) \cdot evalProb(D)$
  end if
```

Algorithm 1 Pseudocode for risk assessment

```
Identify cyclic subsets
$n \leftarrow G_R$ {Begin with graph root node}
$\phi(n) \leftarrow G_V$
$\chi(n) \leftarrow \emptyset$
$\delta(n) \leftarrow \emptyset$
$U \leftarrow G_N - n$ { Initialize set of unevaluated nodes }
while $U \neq \{ \}$ do { some nodes remained unevaluated }
    if $\exists\, n \mid n \in U, \forall p \mid [p, n] \in G_E, p \notin U$ then { node $n$ is ready to be evaluated }
        $P \leftarrow \{p \mid [p, n] \in G_E\}$ { Predecessors of $n$ }
        if $n \in G_D$ then
            $\phi(n) = 1 - evalProb(\overline{P})$
            $\chi(n) = \{ \bigcup_{p \in P} \chi(p)\}$
            $\delta(n) = \{ \bigcap_{p \in P} \delta(p)\}$
            if $n \in G_B$ then
                $\psi(n, n) \leftarrow 1$
            end if
        else { $n \in G_C$ }
            $\phi(n) \leftarrow G_M[n] \cdot evalProb(P)$
            $\chi(n) = (G_B \cap P) \cup \bigcup_{p \in P} \chi(p)$
            $\delta(n) = (G_B \cap P) \cup \bigcup_{p \in P} \delta(p)$
        end if
        $U \leftarrow U - n$ { Mark node $n$ as evaluated }
    else { a cycle is ready for evaluation }
        $C \leftarrow$ cyclic set ready for evaluation (all non-cyclic predecessors evaluated)
        $M \leftarrow evalCycle(C)$
        $U \leftarrow U \setminus M$ { Mark node set $M$ as evaluated }
    end if
end while
```

Algorithm 3 Pseudocode for computing $evalCondProb(D, N)$

```
Require: Parameters $D, N$, such that
    $N = \{n_0, n_1, ..., n_j\}, N \subseteq G_N, |N| \geq 1$, such that $\forall n_i \in N, \phi(n)$ has been evaluated
    $D = \{d_0, d_1, ..., d_k\}, D \subseteq G_N$, such that $D$ d-separates all $n \in N$
    if $\psi(D, N)$ previously calculated then
        return $\psi(D, N)$ from table of stored values
    end if

    if $|N| > 1$ then
        return $\prod_{n \in N} evalCondProb(D, \{n\})$
    else if $N = \{\overline{n}\}$ then {$N$ contains exactly negative element}
        return $1 - evalCondProb(D, \{n\})$

    else { $N = \{n\}$, so $N$ contains exactly positive element }
        $J \leftarrow \{j \mid j \in D\}$ { All positive elements in $D$ }
        $K \leftarrow \{k \mid \overline{k} \in D\}$ { All negative elements in $D$ }

        if $n \in J$ then
            return 1
        end if

        { If $n$ or a dominator of $n$ is negated in $D$ }
        if $n \in K$ or $K \cap \delta(n) \neq \emptyset$ then
            return 0
        end if

        { If set $D$ does not affect the value of $n$ }
        if $D \cap \chi(n) = \emptyset$ then
            return $\phi(n)$
        end if

        $P \leftarrow \{p \mid \exists n \in N, [p, n] \in G_E\}$ {Predecessors of $N$}
        if $n \in G_D$ then
            return $1 - evalCondProb(D, \overline{P})$
        else { $n \in G_C$ }
            return $G_M[n] \cdot evalCondProb(D, P)$
        end if
    end if
```