MEASURING NETWORK SECURITY USING BAYESIAN NETWORK-BASED ATTACK GRAPHS

MARCEL FRIGAULT

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS

SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

MARCH 2010

Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-67114-6
Our file Notre référence
ISBN: 978-0-494-67114-6

ABSTRACT

Measuring Network Security Using Bayesian Network-Based Attack Graphs

Marcel Frigault

Given the increasing dependence of our societies on networked information systems, the overall security of such systems should be measured and improved. Recent research has explored the application of attack graphs and probabilistic security metrics to address this challenge. However, such work usually shares several limitations. First, individual vulnerabilities' scores are usually assumed to be independent. This assumption will not hold in many realistic cases where exploiting a vulnerability may change the score of other vulnerabilities. Second, the evolving nature of vulnerabilities and networks has generally been ignored. The scores of individual vulnerabilities are constantly changing due to released patches and exploits, which should be taken into account in measuring network security. To address these limitations, this thesis first proposes a Bayesian Network-based attack graph model for combining scores of individual vulnerabilities into a global measurement of network security. The application of Bayesian Networks allows us to handle dependency between scores and provides a sound theoretical foundation to network security metrics. We then extend the model using Dynamic Bayesian Networks in order to reason about the patterns and trends in changing scores of vulnerabilities. Finally, we implement and evaluate the proposed models through simulation studies.

Acknowledgments

I sincerely thank Professor Lingyu Wang, my thesis supervisor for all that he has done to ensure my successful completion of this thesis. He has been a source of inspiration through his professionalism and his tireless effort to pursue this field of research. My research with him and our many conversations on various topics have made me appreciate immensely the required dedication and passion required to be a successful academic researcher.

I thank also all of the professors I had during the course portion of this program from whom I learned immensely and drew inspiration to pursue this academic endeavor.

I would also like to thank En Liu who collaborated with me on Polaris and Sirius.

Finally, I thank my wife and four children for their support and understanding during my academic endeavor.

Contents

List of Figures **viii**

List of Tables **xi**

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Thesis Contribution 3

1.3 Thesis Organization 5

2 Literature Review **6**

3 Preliminaries **12**

3.1 Attack Graph 12

3.2 The Common Vulnerability Scoring System (CVSS) 14

3.2.1 Base Scores 14

3.2.2 Temporal Scores 16

3.3 Bayesian Network (BN) and Dynamic Bayesian Network (DBN) 17

3.4 A Probabilistic Approach to Network Security Metric [44] 19

4 **Bayesian Network-Based Attack Graph** **23**

4.1 Individual Score Assignment 23

4.1.1 A Simple Approach 23

4.1.2 A Refined Approach 25

4.2 The Model 29

4.3 Applying the Model 34

5 **Dynamic Bayesian Network-Based Attack Graph** **39**

5.1 The Model 39

5.2 Application 1: Inference of Exploit Node Values 41

5.2.1 An Example 44

5.3 Application 2: Inference of TGS Node Values 47

5.3.1 An Example 49

5.4 Case Study 51

6 **Implementation and Simulation** **61**

6.1 Simulation Environment 61

6.2 Simulation Results 64

7 **Conclusion and Future Work** **70**

Bibliography **71**

List of Figures

1 A Framework for Applying Security Metrics for Hardening Networks . . . 4

2 Network Configuration and Attack Graph 13

3 Propagation of Probability Values 21

4 Example of Transformation Functions 29

5 Case 1 31

6 Case 2 32

7 Case 3 33

8 Case 4 34

9 Case 5 35

10 An Example of Applying the Model 35

11 Conditional Probability Tables for the Left Side Case 36

12 Conditional Probability Tables for the Right Side Case 36

13 Another Example of Applying the Model 37

14 AG Hybrid Model 38

15 Inference of Exploit Node Values 42

16 The DBN Model 45

17 Intraslice CPDs for php (time slice 0) 46

18 Intraslice CPDs for sunvect 47

19 Interslice CPDs for PHP $\tau_{php} = 1/(BS_{php} * TGS_{php})$ 48

20 Interslice CPDs for SunVect Part 1 $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$ 49

21 Interslice CPDs for SunVect Part 2 $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$ 50

22 $P(sunvect) = T$ at Each Time Slice $\tau_{php} = 1/(BS_{php} * TGS_{php})$ and $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$ 51

23 3 Sample Runs $\tau_{php} = 1/(BS_{php} * TGS_{php})$ and $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$ 52

24 Interslice CPDs for PHP $\tau_{php} = 1.25$ 53

25 Interslice CPDs for Sun Vect $\tau_{sunvect} = 1.1$ 54

26 $P(sunvect) = T$ at Each Time Slice $\tau_{php} = 1.25$ and $\tau_{sun} = 1.1$ 55

27 3 Sample Runs $\tau_{php} = 1.25$ and $\tau_{sun} = 1.1$ 56

28 Inference of TGS Node Values 57

29 Interslice CPDs 57

30 The DBN Model 58

31 Inferred Values of E Metric When Exploit and RL Nodes Observed 58

32 DBN for a More Complex AG 59

33 Two Slices 60

34 P(local_bof)=T for Each Time slice 60

35 Attack Graph Generator 62

36 Simulator - Security Metric Calculator (Polaris) 63

37 Simulator - BN Data Analyzer (Sirius) 64

38 BN Size as a Function of AG Size: Normal Distribution 65

39 BN File Size as a Function of AG Size: Normal Distribution 66

40 BN Model Size as a Function of AG Size: Uniform Distribution 67

41 BN Model File Size as a Function of AG Size: Uniform Distribution 68

42 BN Model Size as a Function of AG Size: Initial Conditions 68

43 BN Model File Size as a Function of AG Size: Random Distribution 69

List of Tables

Chapter 1

Introduction

We first introduce the background and motivation of the research. We then summarize the main contributions of this thesis.

1.1 Background and Motivation

Our society has become increasingly dependent on the reliability and proper functioning of a vast number of interconnected information systems. To improve the security of these systems, it is necessary to measure the amount of security provided by different configurations since *you cannot improve what you cannot measure* [19]. The aim of this research is to develop a coherent, logical and applicable security metric for computer networks.

There exists considerable research and standard techniques for measuring individual vulnerabilities, such as the Common Vulnerability Scoring System (CVSS) [7]. such scoring systems typically derive a score based on known facts or experiences about a vulnerability (e.g., whether it can be exploited remotely, whether it requires an authenticated user account, or whether an exploit of the vulnerability is widely available). However, by considering vulnerabilities on an individual basis, a network security administrator could be misled in a situation where the score of each individual vulnerability may be low but these vulnerabilities can be combined to compromise a critical resource which may lead to catastrophic consequences.

More recent research has started to examine quantitative measurements of the overall security of networks. One promising research direction is to employ the model of causal relationships between exploits, namely, *attack graph* (AG) to model the overall security of a network. Existing work along this direction include a general framework [48], a real-valued metric [50], and a probabilistic metric [44]. These works draw strength from both existing security scoring systems of individual vulnerabilities and the attack graph model. More specifically, they combine the measurements of individual vulnerabilities obtained from existing scoring systems into an overall score of the network. Such a combination of scores is based on the causal relationships between vulnerabilities encoded in an attack graph.

However, these existing works share several limitations. First, they usually combine scores of individual vulnerabilities in an arbitrary manner, which prevents them from handling situations where the exploitation of a vulnerability may affect the score of other vulnerabilities. Successfully exploiting a vulnerability at a particular stage of an attack sequence can affect the probability of exploiting a vulnerability at a later stage in the attack sequence. Second, the evolving nature of vulnerabilities and networks has largely been ignored in most existing work. The threat posed by a vulnerability may change over time in today's dynamic network environments. As more technical details of a vulnerability become available, its exploitability or severity may need to be adjusted; when patches are released by vendors to counter an exploit, the vulnerability may become less severe; on the other hand, when exploit codes become more widely disseminated, the severity of a vulnerability may increase. Therefore, it is insufficient to rate vulnerabilities with fixed scores.

To address these limitations, this thesis employs a Bayesian Network (BN)-based attack graph to model the overall security of networks [13] and extends the model using Dynamic Bayesian Networks (DBN) to address the dynamics in network security [14]. More specifically, we first borrow the scores of individual vulnerabilities from the CVSS standard to make the results more applicable in practice. We then show how to interpret attack graphs as special BNs and DBNs and how to combine individual base scores of CVSS based on their causal relationships. Through modeling AGs as special BNs, we provide a sound theoretic foundation for developing probabilistic metrics. In addition, our BN model naturally provides the capability for handling cases where the scores of vulnerabilities may be affected by the exploitation of other vulnerabilities. We extend the BN-based model to a DBN-based model in order to account for the dynamic nature of networks in which vulnerabilities' scores will evolve over time. The DBN model incorporates relevant temporal factors, such as the availability of exploit code or patches, into an attack graph-based security metric. To demonstrate potential applications of the DBN-based model, we present cases where either the Exploitability metric (E) or the temporal score (TS) of a vulnerability is unobservable and can be inferred through reasoning with the model. Finally, we implement and evaluate the proposed models through simulation studies.

Figure 1 shows a framework for applying the security metrics proposed in this thesis in order to improve the security of a network. This thesis will address specifically the generation of annotated AGs, the generation of BN and DBN-based AGs, and the calculation of network security metrics. How to employ the computed metric results to modify a network's current configuration in order to improve its security is a subject of our future work.

1.2 Thesis Contribution

This thesis contributes to the field of network security metrics by proposing two novel models that can quantify network security for complex networks. Specifically, the main

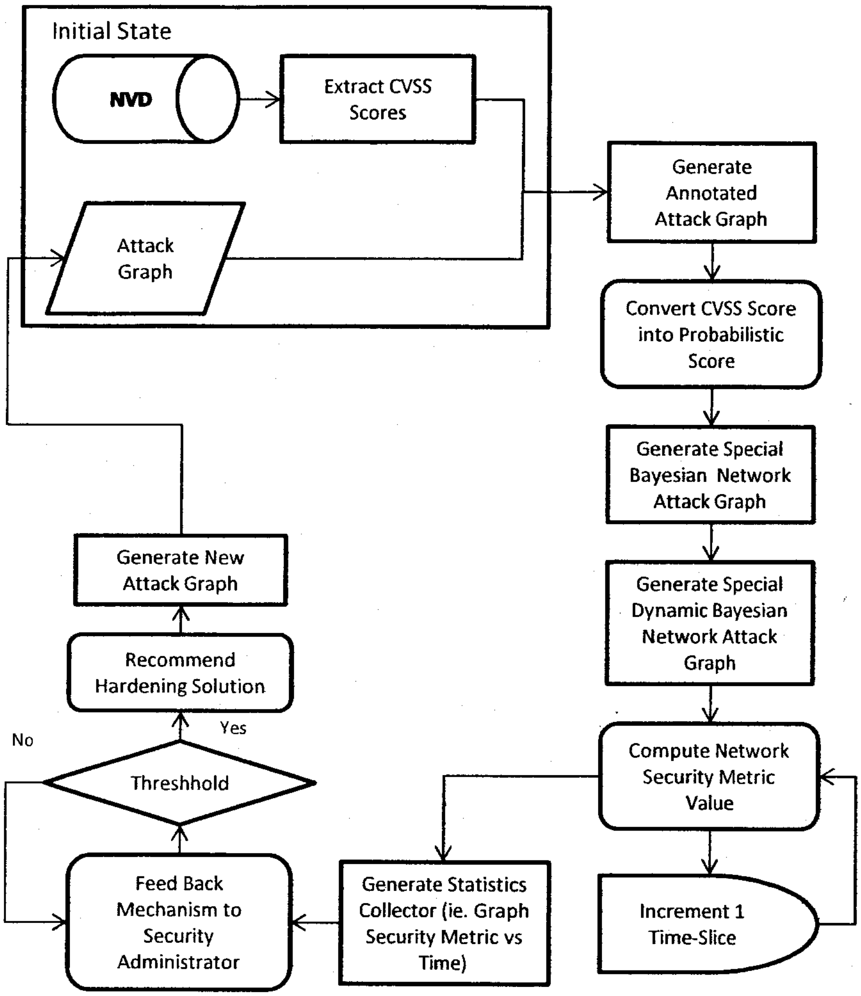

Initial State
NVD
Extract CVSS Scores
Attack Graph
Generate Annotated Attack Graph
Convert CVSS Score into Probabilistic Score
Generate Special Bayesian Network Attack Graph
Generate Special Dynamic Bayesian Network Attack Graph
Compute Network Security Metric Value
Increment 1 Time-Slice
Generate Statistics Collector (ie. Graph Security Metric vs Time)
Feed Back Mechanism to Security Administrator
Threshhold
Yes
No
Recommend Hardening Solution
Generate New Attack Graph

Figure 1: A Framework for Applying Security Metrics for Hardening Networks

contributions are as follows:

- A novel Bayesian Network-based attack graph model is proposed to combine CVSS scores for individual vulnerabilities into a single score for the whole network in a static environment. This model can handle cases where vulnerability scores are not independent and previous models will fail.

- A novel Dynamic Bayesian Network-based attack graph model is proposed to combine CVSS temporal scores into an indicator of dynamic patterns and trends of network security. This is to our best knowledge the first known effort in quantifying the dynamic nature of network security.

- A hybrid model that integrates an existing probabilistic security metric model with the proposed BN-based model is devised. The resulting model will maintain the added capability of the BN-based model while improving the computational efficiency of the model.

- As one of the potential applications, the proposed DBN-based model may provide a sound methodology for refining the CVSS temporal scores.

- Two tools, *Polaris* and *Sirius*, that implement the BN-based models, support the analysis of these models and provide a practical tool for computing network security metrics.

1.3 Thesis Organization

The organization of the remainder of this thesis is as follows. Chapter 2 reviews the state of the art on attack graphs and probabilistic security metrics. Chapter 3 reviews basic concepts of AGs, CVSS, BN and DBNs, and probabilistic network security metric. Chapter 4 describes the BN-based attack graph model and the hybrid model. Chapter 4.1 discusses methods for assigning probabilities to individual vulnerabilities. Chapter 5 describes the DBN-based attack graph model. Chapter 6 presents the Polaris and Sirius tools and simulation results. Finally, chapter 7 discusses future work and concludes this thesis.

Chapter 2

Literature Review

Traditional vulnerability scanners [11] find known vulnerabilities in a network but they cannot reveal how such vulnerabilities can be combined in a multi-step attack to infiltrate a network. To qualitatively evaluate the security of a network against multi-step attacks, a security analyst needs to take into account the effects of interactions between different vulnerabilities and find global security flaws. Traditional approaches to vulnerability analysis of a network typically involve heavy human intervention by the so-called red team. First, vulnerability scanners are used to identify vulnerabilities on individual hosts. Integrating such identified vulnerabilities with other information about the network, such as connectivity between hosts, the red team produces sequences of attacks, namely, attack paths. Each attack path leads to an undesirable state, such as a state where the intruder has administrative accesses to a critical host. The red team approach heavily depends on the skills of the team; the manual process is error-prone, tedious, and not scalable.

Early efforts on the defence against multi-step network attacks exist [8,12,29,51]. The general concept of attack tree is mentioned in [39] as trees with AND and OR for analyzing the security of systems. An attack graph model is described in [33]. A tool is proposed for building an attack graph using forward search in [42]. The inputs of an attack graph include configuration files, attacker profiles, and a database of attack templates manually created; the nodes of the attack graph are attack templates instantiated with particular users and machines; edges are labelled by probabilities of success or cost of attacks. The attack graphs are analyzed to find the shortest paths between given start and end nodes. A *require and provide* approach to automatic attack graph generation is first proposed in [43]. This approach is later widely adopted in defending against multi-step attacks. Attack scenarios can be generated by linking attack steps through their preconditions requirements and post-conditions capabilities. Each successful attack helps attackers to gain more capabilities.

Model checking is first applied to the analysis of multi-step network attacks in [35] where known vulnerabilities on network hosts, connectivity between hosts, and the initial capabilities of the attacker together form states, whereas exploits form transitions between states executed by attackers. Such a formal model is given to a model checker as the input while the reachability in terms of given goal states as a query. The model checker will produce a counterexample if there exists a sequence of exploits leading to the goal states. Such a sequence of exploits indicates a potential attack path that must be broken in order to secure the network. The authors of [36] provide more details on how connectivity should be modeled at different layers and the term *topological vulnerability analysis* is introduced. Later, model checking is used differently in [20, 40], that is, to enumerate all attack paths. A modified model checker applied to the finite-state machine created from network information provides all counterexamples to a query stating the safety of goal states, which are essentially the collection of possible attack paths. Analysis is possible on such a model, such as finding a *cut set* in the attack graph so that goal conditions can no longer be reached. The problem of finding the minimum attack leading to given goal conditions is shown to be intractable.

To address the scalability issue of model checking-based approaches, a *monotonic assumption* is adopted in [1] that states the further exploits will never cause the attacker to relinquish any obtained privileges. Attack paths can be implicitly represented as paths in a directed graph; the latter includes exactly one copy of each exploit and its pre- and post conditions, with edges to interconnect exploits to these conditions. This assumption reduces the complexity of attack graphs from exponential in the number of hosts to polynomial. It may also cause some attacks that may disable services or invalidate vulnerabilities impossible to be included in the model. Attack graphs can be created with a two-pass search; the first connects exploits by starting from the attacker's initial state, and then prunes those irrelevant states by searching backward from the goal state. Other analyses are also possible, such as finding minimal attacks leading to given goal conditions. More recently, a logic programming-based approach to attack graphs is given in [31]. The Datalog language is used to encode knowledge about attacks in a network. MulVAL [30] is a security analyzer built from off-the-shelf tools, which can be used to retrieve information regarding software and vulnerabilities. The engine takes as input the network configuration information and outputs attack steps leading to the compromise of the network. The analysis has polynomial complexity in the size of the network.

A treatment of the scalability issue of attack graph representation is given in [37]. A hierarchical approach builds rules at every level of aggregation and then integrates them through common attribute values of attack graph elements or attack graph connectivity. Attack subgraphs are recursively collapsed into single vertices so that compression is possible to a certain degree. Moreover, the abstraction of protection domains is proposed to reduce complexity when groups of machines have complete connectivity. A quadratic complexity is claimed. Another effort applies a matrix clustering algorithm to the adjacency matrix of attack graphs so the resulting adjacency matrix indicates the feature of protection domain on the main diagonal [38]. Two other improvements to the representation of attack graphs are given in [16]. A directed graph is used to model subnets as nodes and potential inter-subnet attacks as edges. A dominator tree is used to determine whether inter-subnet and intra-subnet attacks are possible based on the domination relationships. Then, an abstraction reduces groups of exploits to virtual nodes so as to increase the readability of the attack graph. Both methods may reduce the complexity of visualized attack graphs and allow human users to quickly grasp imminent threats.

Recently, much interest has focused on quantifying the threat of potential multi-step attacks. General reviews of security metrics are given in [4, 19, 25]. The NIST's efforts on standardizing security metrics are given in [27] and more recently in [41] and in the Common Vulnerability Scoring System (CVSS) [7]. Another overview of many aspects of network security metrics is given in [17]. Dacier et al. gave intuitive properties that should be satisfied by any security metric [8, 9, 29]. Based on the exploitability concept, a qualitative measure of risk is given in [5]. The difficulty of attacks are measured in terms of time and efforts spent by attackers. Founded on an exponential distribution for an attacker's success rate over time, they use the Markov model and the MTTF (Mean Time to Failure) to measure the security of a network. They discussed simple cases of combining individual measures but did not study the general case. Another approach measures the relative risk of different configurations using the *weakest attacker* model, that is the least conditions under which an attack is possible [32]. Yet another series of work measures how likely a software is vulnerable to attacks using a metric called *attack surface* [24]. These works allow a partial order to be established on different network configurations based on their relative security. The work by Balzarotti et al. [5] focuses on computing the minimum efforts required for executing each exploit. However, the treatment of many aspects of security is still qualitative in nature. For example, the resources are still treated equally important (no explicit evaluation of damages) and the resistance to attacks is regarded as binary (an attack is either impossible or trivial).

Relevant work exists in other areas, such as the study of trust in distributed systems. Beth et al. proposed a metric for measuring the trust in an identity that has been established through overlapping chains of certificates [6]. The way they combine values of trust in certificates into an overall value of trust proves to be useful in the study of security metrics. Similarly, the design principles given by Reiter et al. are intended for developing metrics of trust and we found these principles applicable to our study [34]. Structures similar to attack graphs are used for risk analysis in safety-critical systems although the focus is not on vulnerabilities but on trust relationships [3]. Our model, used as a monitoring system, shares similarity with the techniques for testing whether a finite execution of events generated by a program violates a linear temporal logic (LTL) formula [15]. To generate attack graphs, topological vulnerability analysis enumerates potential multi-step intrusions based on prior knowledge about vulnerabilities and their relationships [35, 40]. On the attack response front, attack graphs have been used for the correlation of attacks, the hypotheses of alerts missed by IDSs, and the prediction of possible future attacks [45,46,50].

Wang et al. [48] proposed a framework for using combining functions to determine the combined effect of vulnerabilities in a network. They proposed the idea of using an analogy to the resistance of electrical circuits in [49] and address the issue of additional dependency between exploits although the solution is not entirely satisfactory since cycles in attack graphs are largely ignored. Wang et al. also proposed a probabilistic network security metric based on attack graphs [13, 14, 44]. This work proposes the use of probability scores for each vulnerability to represent the likelihood that one attacker will exploit the vulnerability or the percentage of attackers that successfully exploit the vulnerability. Our work adopts this same concept but uses it to develop conditional probability tables for each exploit and then demonstrates how the use of BNs can be used to determine network security. The work on minimum-cost network hardening represents an early effort toward the quantitative study of network security [47]. This work quantifies the cost of removing vulnerabilities in hardening a network, but it does not consider other hardening options, such as modifying the connectivity. It also has the limitation of adopting a qualitative view of damages (that is, all the given critical resources are equally important) and of attack resistance (that is, attacks on critical resources are either impossible or trivial).

The idea of using BNs to model network vulnerabilities and determine a quantitative value representing the security of a network has been explored by Liu and Man [23]. A BN is used to model all potential atomic attack steps in a network. Each vertex represents a single security property violation state and each edge corresponds to an exploitation of one or more exhibited vulnerabilities. They assign edge weights to represent the probability of successful exploits. The difference between their work and ours will be detailed later in this thesis. Our application of DBN is inspired by the work of An et al. [2] for privacy intrusion detection. They employ DBN to relate a database operator's intention to observable factors, such as the time spent on a certain operation. Our work is based on similar ideas but applies the concept of DBN model to combining the CVSS scores for measuring network security.

Chapter 3

Preliminaries

To be self-contained, this chapter reviews the relevant concepts of attack graphs, the CVSS standard, BNs, DBNs, and a previous approach to a probabilistic network security metric.

3.1 Attack Graph

An attack graph (AG) models our knowledge about how multiple vulnerabilities may be combined for an attack. The model represents system states using security-related conditions, such as the existence of vulnerabilities on a host or the connectivity between hosts, and state transitions using exploits of vulnerabilities. For the purposes of this thesis, an AG is a directed graph with conditions and exploits as vertices, and their relationships as edges. More formally, we have the following.

Definition 1 *An attack graph G is a directed graph $G(E \cup C, R_r \cup R_i)$ where E is a set of exploits, C a set of conditions, and $R_r \subseteq C \times E$ and $R_i \subseteq E \times C$.*

The left-hand side of Figure 2 depicts a simple scenario where a file server (host 1) offers the File Transfer Protocol (ftp), secure shell (ssh), and remote shell (rsh) services; a database server (host 2) offers ftp and rsh services. The firewall only allows ftp, ssh, and rsh traffic from a user workstation (host 0) to both servers. The right-hand side depicts the AG in which exploits of vulnerabilities are depicted as predicates in ovals and conditions as predicates in clear texts. The two numbers inside the parentheses denote the source and destination hosts, respectively. The AG represents three self-explanatory sequences of attacks (attack paths). For example, the right path is: $sshd\_bof(0,1) \rightarrow ftp\_rhosts(1,2) \rightarrow rsh(1,2) \rightarrow local\_bof(2)$.

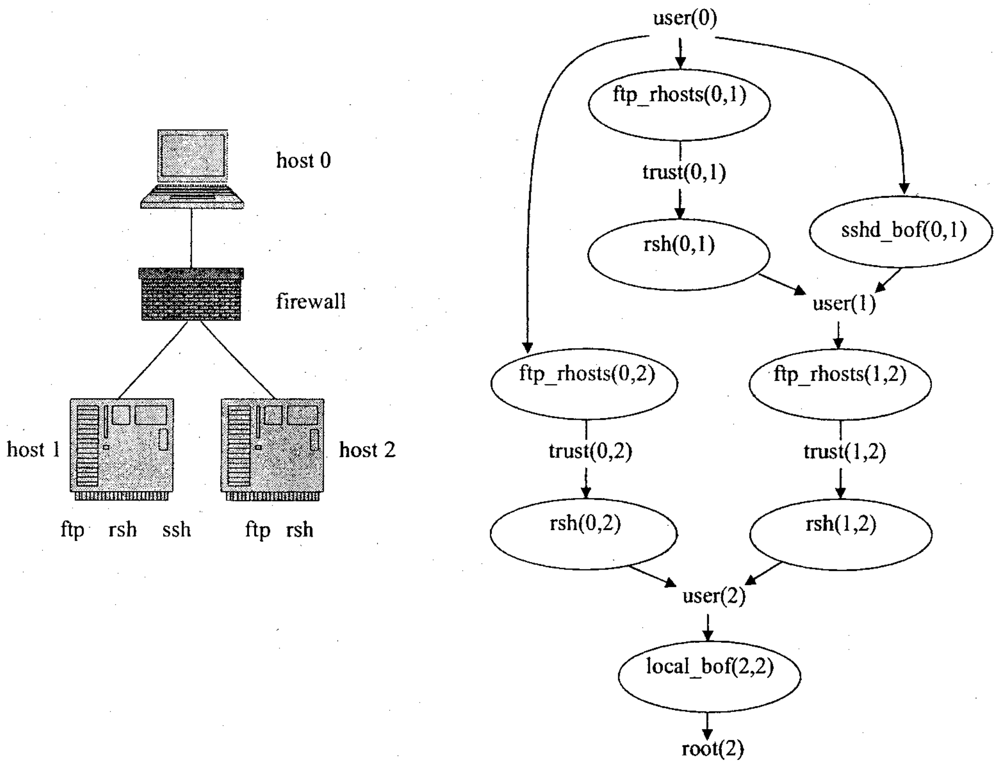


Figure 2: Network Configuration and Attack Graph

We make two assumptions. First, the AG of a given network can be obtained using existing tools, such as the Topological Vulnerability Analysis (TVA) system, which can generate AGs for more than 37,000 vulnerabilities taken from 24 information sources including X-Force, Bugtraq, CVE, CERT, Nessus, and Snort [18]. Second, the CVSS scores of vulnerabilities in the given AG can be obtained from existing vulnerability databases, such as the National Vulnerability Database (NVD) [28].

3.2 The Common Vulnerability Scoring System (CVSS)

The models proposed in subsequent sections will borrow scores assigned to individual vulnerabilities according to the Common Vulnerability Scoring System (CVSS) [7]. The CVSS is an open and free framework that provides a means of assigning quantitative values to vulnerabilities based on several metrics. The metrics are divided into three categories: *Base Scores, Temporal Scores and Environmental Scores.* For the purposes of the work presented in this thesis, only the Base and Temporal metric concepts will be considered.

3.2.1 Base Scores

The *Base Score* (BS) for each vulnerability quantifies its intrinsic and fundamental properties that are supposed to be constant over time and independent of user environments. The Base Score ranges from 0 to 10 and is calculated based on the following six metrics:

- Access Vector - AV: This indicates the types of accesses required for exploiting the vulnerability. Possible values are Local (numerical value 0.395), Adjacent Network (0.646), and Network (1.0), which are all self-explanatory.

- Access Complexity - AC: A quantitative measure of the attack complexity required to exploit the vulnerability. The range of values are: High (0.35), Medium (0.61) and Low (0.71).

- Authentication - Au: A measure of the the number of times an attacker must authenticate to a target in order to exploit a vulnerability. The defined range of values are: Multiple (0.45), Single (0.56) and No (0.704).

- Confidentiality - C: A measure of the impact on confidentiality following a successful exploitation with the following defined range of values: None (0.0), Partial (0.275) and Complete (0.660).
- Integrity - I: A measure of the impact on integrity following a successful exploitation with the following defined range of values: None (0.0), Partial (0.275) and Complete (0.660).
- Availability - A: A measure of the impact on availability following a successful exploitation with the following defined range of values: None (0.0), Partial (0.275) and Complete (0.660).

The CVSS Framework imposes the use of a vector which encodes the metric score values used to compute the overall score for a vulnerability. The following is an example vector:

$$AV:N/AC:L/Au:N/C:N/I:C/A:C$$

from which we can derive the numerical scores as indicated above.

The Base Metric score (BS) is computed as follows:

- $Impact = 10.41 * (1 - (1 - ConfImpact) * (1 - IntegImpact) * (1 - AvailImpact))$
- $Exploitability = 20 * AccessVector * AccessComplexity * Authentication$
- $f(impact) = 0$ if $Impact = 0$, 1.176 otherwise
- $BaseScore = round\_to\_1\_decimal((0.6 * Impact) + (0.4 * Exploitability) - 1.5) * f(Impact))$

Using the example vector, the following demonstrates how to compute the BS:

- $Exploitability = 20 * 1 * 0.71 * 0.704 == 9.9968$

- $Impact = 10.41 * (1 - (1 - 0) * (1 - 0.660) * (1 - 0.660) == 9.2066$
- $f(impact) = 1.176$
- $BaseScore = round\_to\_1\_decimal((0.6{*}9.2066){+}(0.4{*}9.9968){-}1.5){*}1.176 ==$ 9.4

3.2.2 Temporal Scores

The *Temporal Score* (TS) quantifies a vulnerability when considering properties of the vulnerability that may change over time. The three temporal metric values used in CVSS are:

- Exploitability - E: Indicates the current state regarding the availability of exploit codes or techniques, with the following defined range of values: Unproven (0.85), Proof-of-concept (0.90), Functional (0.95), High (1.00) and Not Defined (1.00).
- Remediation Level - RL: Indicates the current situation regarding the availability of remediation solutions. The defined range of values is: Official Fix (0.87), Temporary Fix (0.90), Workaround (0.95), Unavailable (1.00) and Not Defined (1.00).
- Report Confidence - RC: Indicates the degree of confidence regarding the existence of a vulnerability and the technical details. The range of defined values is: Unconfirmed (0.90), Uncorroborated (0.95), Confirmed (1.00) and Not Defined (1.00).

The Temporal Metric Score (TS) is computed as follows:

$$TS = round\_to\_1\_decimal(BS * E * RL * RC) \tag{1}$$

For convenience, the following product is defined:

$$TGS = (E * RL * RC) \tag{2}$$

$$TS = round\_to\_1\_decimal(BS * TGS) \tag{3}$$

Consider the example of a vulnerability with the above Base Score and the following Temporal vector:

$$E : POC/RL : W/RC : C$$

The Temporal Score (TS) is calculated as follows:

- $TGS = 0.90 * 0.95 * 1 == 0.855$
- $TS = 9.4 * 0.855 == 8.0$

3.3 Bayesian Network (BN) and Dynamic Bayesian Network (DBN)

BNs offer a compact means to encode the entire range of conditional relationships in the system being modeled. A BN can be defined as a directed acyclic graph (DAG) with nodes representing variables and arcs representing conditional independencies among the variables [26]. More formally, the BN for a system X can be formally described as a pair $B = (G, Q)$ where G is a DAG and Q is the set of parameters that quantify the network, such as the conditional distribution values for each variable (node). The joint distribution for a BN is represented by:

$$P(X_1...X_n) = \prod_{i=1}^{n} P(X_i|parents(X_i)) \tag{4}$$

In [44], the notion of assigning to each node of an AG a probability value that represents the likelihood that one attacker or the percentage of attackers that will exploit the vulnerability was discussed. However, they do not employ BN as the basis of their model. In this thesis, the same AG annotated with the individual probabilities will be used, but the Conditional Probability Tables (CPT) for each node of the AG will be developed. The AG, represented as a DAG, coupled with the CPT that encodes the conditional independencies for all nodes will constitute a BN.

It is important to distinguish the BN-based AG used in this thesis from that of Liu et al. [23]. Liu et al. represent each node of the graph as a host with a specific security violation state, whereas in the AGs of this thesis, each node represents vulnerabilities as well as the pre- and post-conditions resulting from the exploitation of such vulnerabilities. Liu et al. assign probabilities to the edges, whereas this thesis assigns them to nodes. While the probabilities for nodes can be readily obtained from widely available standard measures such as CVSS, we believe probabilities to be assigned to edges are harder to obtain in practice.

In contrast to BNs, DBNs are graphical models for probabilistic inferences in dynamic domains that can enable users to monitor and update the system as time proceeds, and even predict further behaviors of the system [26]. Today's networks are certainly dynamic environments, and the security of such environments involves many temporal factors, such as the availability of exploit code, the availability of patches or fixes, the confidence in reported vulnerabilities, and so on. To incorporate such temporal factors in measuring network security, this thesis extends the BN-based model to DBNs.

In a typical DBN model, the system is represented as a sequence of BNs. Each BN represents a time *slice* of the DBN corresponding to a particular instant of time. As with the BN, arcs exists between the vertices within each time slice. In addition, the DBN will have arcs between certain vertices of successive time slices. For simplicity, it is generally assumed that a DBN satisfies the Markovian property which implies that the state of the system depends only on the previous state. In addition, it is assumed that the conditional dependencies among the vertices across the time slices are the same. Therefore, the system can be modeled with only 2 time slices (more strictly speaking, the first 1.5 slices).

In DBNs, the vertices can be classified as either *observable* or *unobservable*. The value of observable vertices are known apriori to the analysis process, whereas the values of the unobservable variables are not available but may be inferred. In order to provide the required links between the time slices, arcs can be introduced between a set of unobservable vertices and the necessary conditional probability distributions can be developed to encode the relationships existing between successive time slices.

3.4 A Probabilistic Approach to Network Security Metric [44]

We briefly introduce an existing probabilistic approach to combining scores of individual vulnerabilities [44], which will be required in later discussions. With this approach, the events that an attacker can (and will) execute different exploits will be assumed as independent, and removing such an assumption will be an important contribution of this thesis. Also, a fixed probability for measuring vulnerabilities is assumed.

The approach will associate each exploit e and condition c with two probabilities, namely, $p(e)$ and $p(c)$ for the *individual score*, and $P(e)$ and $P(c)$ for the *cumulative score*. The individual score $p(e)$ stands for the intrinsic likelihood of an exploit e being executed, given that all the conditions required for executing e in the given attack graph are already satisfied. On the other hand, the cumulative score $P(e)$ and $P(c)$ measures the overall likelihood that an attacker can successfully reach and execute the exploit e (or satisfy the condition c) in the given attack graph.

For exploits, the individual score is assigned based on expert knowledge about the vulnerability being exploited. For conditions, it is assumed in this approach that the individual score of every condition is always 1. Intuitively, a condition is either initially satisfied or immediately satisfied after a successful exploit (in practice, we can easily remove such assumptions by assigning less-than-1 individual scores to conditions). It is proposed that individual scores can be obtained by converting vulnerability scores provided by existing standards, such as the CVSS base score and temporal score [7], to probabilities, although no details are discussed.

Unlike individual scores, the cumulative score takes into account the causal relationships between exploits and conditions. In an attack graph, such causal relationships may appear in two different forms. First, a conjunction exists between multiple conditions required for executing the same exploit. Second, a disjunction exists between multiple exploits that satisfy the same condition. The cumulative scores are defined in the two cases similar to the probability of the *intersection* and *union* of random events. That is, if the execution of e requires two conditions c_1 and c_2, then $P(e) = P(c_1) \cdot P(c_2) \cdot p(e)$; if a condition c can be satisfied by either e_1 or e_2 (or both), then $P(c) = p(c)(P(e_1) + P(e_2) - P(e_1) \cdot P(e_2))$. Definition 2 formalizes cumulative scores.

Definition 2 *Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$, and any individual score assignment function $p : E \cup C \to [0, 1]$, the cumulative score function $P : E \cup C \to [0, 1]$ is defined as*

- $P(e) = p(e) \cdot \prod_{c \in R_r(e)} P(c)$
- $P(c) = p(c)$*, if* $R_i(c) = \phi$*; otherwise,* $P(c) = p(c) \cdot \oplus_{e \in R_i(c)} P(e)$ *where the operator* $\oplus$ *is recursively defined as* $\oplus P(e) = P(e)$ *for any* $e \in E$ *and* $\oplus(S_1 \cup S_2) = \oplus S_1 + \oplus S_2 - \oplus S_1 \cdot \oplus S_2$ *for any disjoint and non-empty sets* $S_1 \subseteq E$ *and* $S_2 \subseteq E$.

Figure 3 illustrates the AG from Figure 2 but only includes the exploit nodes. The individual probability scores for each node are shown in plaintext beside each node and the cumulative score is shown in parentheses.

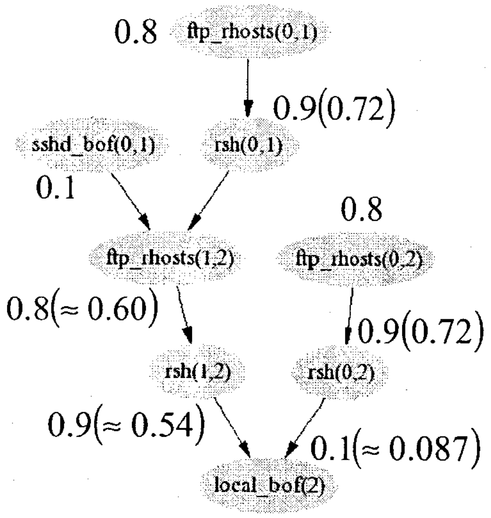


Figure 3: Propagation of Probability Values

The cumulative scores of two exploits can be calculated as follows.

1. $P(rsh(0,1)) = P(trust(0,1) \times p(rsh(0,1)) = 0.8 \times 0.9 = 0.72$
2. $P(user(1)) = P(rsh(0,1)) + P(sshd\_bof(0,1)) - P(rsh(0,1)) \times P(sshd\_bof(0,1)) = 0.72 + 0.1 - 0.72 \times 0.1 = 0.748$

The following can now be proposed as a definition of a network security metric:

Definition 3 *Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$, any individual score assignment function $p : E \cup C \to [0,1]$, a cumulative score function $P : E \cup C \to [0,1]$ and a goal state $g \in C$, the overall network security metric value (SM) is the probability*

$$SM = P(g = True) \tag{5}$$

Using probabilities for a security metric has been criticized as violating a basic design principle, that is, the value assignment should be specific and unambiguous rather than abstract and meaningless [34]. However, there is a simple interpretation for the metric proposed in this thesis. That is, the individual score $p(e)$ is the probability that any attacker can, and will execute e during an attack, given that all the preconditions are already satisfied. Equivalently, among all attackers that attempt to compromise the given network during any given time period, $p(e)$ is the fraction of attackers that can, and will execute e.

This interpretation of individual scores considers two factors in determining the individual score $p(e)$, namely, whether an attacker has the skills and resources to execute e and whether he/she will choose to do so. For example, a vulnerability that cannot be exploited remotely, or one that requires a valid user account will likely have a lower score due to the first factor (that is, fewer attackers *can* exploit the vulnerability), whereas a vulnerability that can be easily detected, or one less exposed to the public will likely have a lower score due to the second factor (that is, fewer attackers *will* exploit the vulnerability).

The interpretation of individual scores also provides a natural semantics to the cumulative scores. That is, $P(e)$ or $P(c)$ stands for the likelihood, or the fraction of, attackers who will successfully exploit e or satisfy c in the given network. The cumulative score of a given goal condition thus indicates the likelihood that a corresponding resource will be compromised during an attack, or equivalently, among all attackers attacking the given network over a given time period, the average fraction of attackers who will successfully compromise the resource. Such a likelihood or fraction is clearly relevant in analyzing the security of a network or in hardening the network for better security.

Chapter 4

Bayesian Network-Based Attack Graph

This chapter proposes a BN-based attack graph approach to combining individual scores as a measure of the overall security of a network. We first discuss the assignment of individual scores, and then introduce the model and apply it to several case studies.

4.1 Individual Score Assignment

In this section, we discuss how individual scores can be assigned with two different approaches.

4.1.1 A Simple Approach

An attack graph enumerates all possible sequences of vulnerability exploitations leading to a goal state. However, an AG is still qualitative in nature, and does not directly provide a way to measure the security of a network. The approach we take in this thesis is to first derive probability values for each exploitation of a vulnerability, and then use these values to annotate the AG. The resulting graph will be referred to as an *annotated attack graph.* The BN and DBN models developed later in this thesis are based on such an annotated attack graph. We base the assignment of individual scores upon the CVSS standard [7] since it is widely accepted and the CVSS scores are readily available from many resources, such as the NVD [28].

The first approach simply converts CVSS scores of a vulnerability to probabilities as follows. First, we convert the score BS (or TS in the dynamic case) to a probability using a simple approach of diving it by the domain size 10. We then associate this probability to all the exploits that have this vulnerability (recall that an exploit is a vulnerability bound to specific source and destination hosts). CVSS scores are proposed for quantifying individual vulnerabilities only and ignore the causal relationships between exploits in the context of a given network, which is modeled in attack graphs. Therefore, we define the probability converted from a score as the conditional probability of an exploit when all of its preconditions in the attack graph are already satisfied (by other exploits that imply those conditions).

More formally, consider an attack graph G as a directed graph $G(E \cup C, R_r \cup R_i)$ where E is a set of exploits, C a set of conditions, and $R_r \subseteq C \times E$ and $R_i \subseteq E \times C$ are two relations. The approach of this thesis regards each exploit as a binary variable that can take discrete values of T (True), which signifies the exploit has been successfully performed by the attacker, or F (False) indicating the converse. Given any exploit $e \in E$, and its corresponding score BS (or TS in the dynamic case), this proposal defines the Individual Score Assignment Function used to assign the individual probability scores to each exploit node of G as follows:

Definition 4 *Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$ and $e = T, F$ the Individual Score Assignment Function $p : E \cup C \to [0, 1]$ can be defined as follows:*

$$p(e = T | \forall c \in R_r(e)\ c = T) = BS_e/10 \tag{6}$$

For example, in Figure 2, we have $P(rsh(0,1) = T | trust(0,1) = T) = BS_{rsh(0,1)}/10$.

Since the condition $trust(0, 1)$ can only be satisfied by one exploit $ftp\_rhosts(0, 1)$, the model can relate probabilities of the two exploits as $P(rsh(0, 1) = T|trust(0, 1) = T) = P(rsh(0, 1) = T|ftp\_rhosts(0, 1) = T) = BS_{rsh(0,1)}/10$.

4.1.2 A Refined Approach

With the approach discussed in the previous section, we are essentially regarding the CVSS base score assignment scheme as a blackbox. In this section, we shall take a closer look at the internal structure of this scheme, and refine our previous approach accordingly.

As explained in Section 3.2, the CVSS Base Score is assigned based on metrics AV, AC, Au, C, I, A. We discuss how these metrics would affect our individual score assignment as follows.

- AV: The more remote an attacker can be from the target machine, the more likely a vulnerability will be exploited since the number of potential attackers will be greater.
- AC: The lower the required access complexity, the greater the likelihood of attacks becomes.
- Au: The stronger the authentication required for exploiting a vulnerability is, the lesser the likelihood of attacks is.
- C, I, A: It can be argued that these three impact metrics have no direct relevance to the determination of the likelihood of attacks since they measure the aftereffect. However, they may in fact affect the likelihood of exploitation in the following sense. The greater the potential impact is, more attractive the vulnerability will be to an attacker. Therefore, we believe that these impact metrics are also relevant to the assignment of individual scores.

Therefore, the individual score, or the probabilistic values representing the likelihood of exploiting any particular vulnerability v can be represented as a function of the six base metric component scores:

$$P(v = T) = f(AV, AC, Au, C, I, A) \tag{7}$$

In the previous section, we have implicitly defined $f(AV, AC, Au, C, I, A)$ using the equations provided within CVSS (with the division by 10). Upon a closer look, this approach has the following limitations. It cannot model the effect of an exploitation upon subsequent exploitations of the same vulnerability. Logically, an attacker would employ previously gained knowledge, skills, and tools to facilitate subsequent exploitations and thus the probability of the latter would be greater. Moreover, other types of dependency may also exist between exploitations, and the effect of such dependency may vary between different base metric component scores. More specifically,

- AV: Consider a situation in which multiple exploits exist on a single host and an attacker must exploit them in a specific sequence to attain his objective. Clearly, if some exploits require the same type of accesses, then we should not count such accesses more than once since a gained access right will remain with the attacker. Therefore, if the attacker has already gained the access required from an earlier exploit, we should compute an adjusted value for the current exploit, instead of simply using the one provided by CVSS scores.

- AC: Similarly, in situations where multiple exploits exist on a single host and an attacker must exploit them in a specific order, some exploits may reduce the access complexity of subsequent exploits, and we should compute an adjusted value for the latter instead of using the one provided by CVSS.

- Au: In this case, authentication measures required by different exploits form a hierarchy. Certain exploits may require an authentication method that implies that an attacker will have the required credentials for subsequent exploits, an adjusted value should be used for the latter.

- C, I, A: We assume the C, I, and A metric scores of a vulnerability remains unaffected by successful exploitations.

To compute adjusted probability scores for each AG node, two approaches are possible:

- Option 1: Compute adjusted base score component metric values for each node of the AG and then reuse the CVSS equations to recompute the base score metric (then divide by 10).

- Option 2: Define new equations to compute the probability scores.

In this thesis, we will adopt the first approach and regard the second as future work. More specifically, we define transformation functions for the adjusted base metric component scores:

- $AV_e' = \alpha(AV_e)$: This transformation function adjusts the initial AV metric value from CVSS by considering the network characteristics as follows. All predecessor AG nodes (or ancestors) on the same host are examined through a backward search. If a node has an AV metric equal to, or implies the access requirement of the node being analyzed, then the transformation function should result in $AV' \geq AV$. Later in this section, we will describe an example of such transformation functions.

- $AC_e' = \beta(AC_e)$: This transformation function adjusts the AC metric value from CVSS by determining to what degree the attacker has acquired knowledge from exploiting previous vulnerabilities that will make this new exploitation easier (with less complexity). Again, we will discuss an example later in this section.

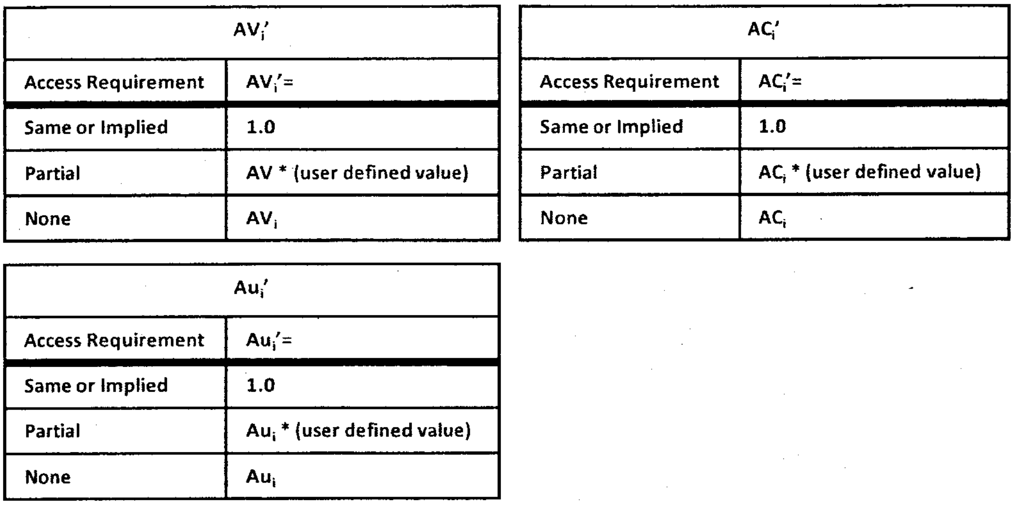
- $Au'_e = \pi(Au_e)$: This transformation function adjusts the initial Au metric value obtained from CVSS by considering if the attacker has already provided the same, or implied, authentication credential during a previous exploit.

Definition 5 *Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$, the individual score assignment function $p : E \cup C \rightarrow [0, 1]$ is defined as*

$$p(e = T | \forall c \in R_r(e)\ c = T) = f(\alpha(AV_e), \beta(AC_e), \pi(Au_e), C_e, I_e, A_e) \tag{8}$$

Consider a simple example with two computers, host 0 and host 1, and assume that an attacker has user access to host 0 (namely, condition $c1(0)$) and network connectivity to host 1. Also, suppose host 1 is running services with vulnerabilities $v1$, $v2$ and $v3$. $v1$ can be exploited from host 0 to host 1 resulting in the post-condition $c2(0, 1)$; $v2$ can then be exploited resulting in $c3(1)$; finally, $v3$ can be exploited, which is the goal state for the attacker.

In this simple case, the base metrics component scores obtained from CVSS (divided by 10) should be assigned to $v1(0, 1)$ since it is the first exploit. For exploit $v2(0, 1)$ (and similarly $v3(0, 1)$), if the same access vector, access complexity or type of authentication is already required by $v1(0, 1)$, then we set the value of AV, AC, or Au to be 1, which is the greatest possible value, to indicate a dependency between these exploits. If $v1(0, 1)$ only meets partially the access requirements of $v2(0, 1)$, then we can multiply the CVSS scores for $v2(0, 1)$ (and similarly for $v3(0, 1)$) by a user-defined value. Finally, if the access requirements of $v1(0, 1)$ are completely different from those of $v2(0, 1)$ (and $v3(0, 1)$), then we should directly use the CVSS scores for the latter. This example of transformation functions is summarized in Figure 4.



AV_i'	
Access Requirement	AV_i'=
Same or Implied	1.0
Partial	AV * (user defined value)
None	AV_i

AC_i'	
Access Requirement	AC_i'=
Same or Implied	1.0
Partial	AC_i * (user defined value)
None	AC_i

Au_i'	
Access Requirement	Au_i'=
Same or Implied	1.0
Partial	Au_i * (user defined value)
None	Au_i

Figure 4: Example of Transformation Functions

4.2 The Model

Our model takes as input an annotated AG $G(E \cup C, R_r \cup R_i)$ where each vertex is annotated with a probability, or individual score, assigned according to Equation 6. We generate a corresponding BN-based attack graph $B = (G, Q)$. G is a directed graph corresponding to the AG but with different semantics, that is, the vertices represent the binary variables of the system and the edges represent the conditional relationships among the variables. Q is the set of parameters that quantify the BN including the conditional distribution values for each variable. The CPD tables can then be developed to propagate probabilities, or cumulative scores, along the AG until reaching the goal condition.

The unique aspect of this BN representation is the following. In an AG, the causal relationships between exploits can be disjunctive or conjunctive based on how they are related through conditions [13]. Such relationships are represented in the BN representation through special conditional probabilities of 0 or 1. More specifically,

- We say a *disjunctive* relationship exists between any exploits $e_1, e_2, \ldots, e_n$ with respect to e_{n+1} when $e_j R_i c$ holds for all $j = 1, 2, \ldots, n$ and some condition c, and

$cR_r e_{n+1}$ is true. In such a case, the probability assignment based on Equation 6 will satisfy $P(e_{n+1} = T|X) = 1$ for all X that has $e_j = T$ hold for at least one $j \in [1, n]$.

- We say a *conjunctive* relationship exists between exploits $e_1, e_2, \ldots, e_n$ with respect to e_{n+1} when $e_j R_i c_j$ and $c_j R_r e_{n+1}$ both hold for all $j = 1, 2, \ldots, n$ and some conditions c_j's. In such a case, we have $P(e_{n+1} = T|X) = 0$ whenever X has $e_j = F$ hold for at least one $j \in [1, n]$.

We illustrate the above concept through the study of several special cases. Presently there exists no standard against which to evaluate the accuracy of the BN model proposed in this thesis given the short history of research in this field and the lack of availability of experimental data. One method that is available to provide a degree of evaluation of a security metric model's accuracy is to establish how adequately the model handles intuitive properties that any security metric should satisfy. The development of a comprehensive list of such intuitive properties is left as a subject for future research, however, it will be demonstrated in the following cases that the BN model proposed in this thesis can handle a set of intuitive properties that any security metric should satisfy.

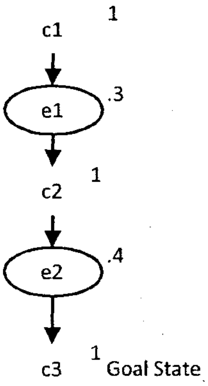
Case 1: $e1$ and $e2$ must be exploited in the given order.

In Figure 5, each node has been annotated with an individual score. As previously mentioned, these scores are assigned based on CVSS and the scores for conditions are set to 1. Next, the CPT is generated for each node to encode the cumulative score. In this model, the nodes are assigned discrete values of T indicating that the exploit has been successfully executed, or F indicating otherwise. For conditions, T indicates that the condition has been satisfied, and F otherwise. The CPT allows for the calculation of the joint probability function. The objective is to determine the probability of satisfying the goal condition

$c3 = T$. This can be calculated as follows:

$$P(c3 = T) = \sum_{e2,c2,e1,c1\epsilon\{T,F\}} P(c3 = T, e2, c2, e1, c1)$$

$$= 0.12$$



e1		
c1	F	T
F	1	.7
T	0	.3

e2		
c2	F	T
F	1	.6
T	0	.4

c1	
F	0
T	1

c2		
e1	F	T
F	1	0
T	0	1

c3		
e2	F	T
F	1	0
T	0	1

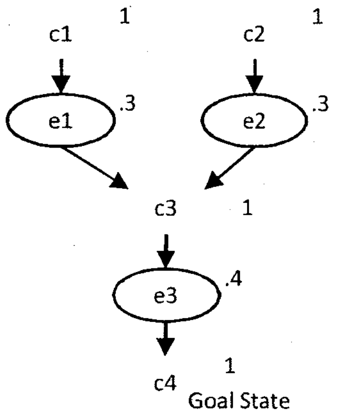
Figure 5: Case 1

Case 2: Either $e1$ or $e2$ must be exploited.

Figure 6 shows the second case where disjunctive relationship exists between two exploits. The probability of satisfying the goal condition $c4 = T$ can be calculated as follows.

$$P(c4 = T) = \sum_{e3,c3,e1,e2,c1,c2\epsilon\{T,F\}} P(c4 = T, e3, c3, e1, e2, c1, c2)$$

$$= 0.204$$



C1	
F	0
T	1

c2	
F	0
T	1

C3				
e1	F		T	
e2	F	T	F	T
F	1	0	0	0
T	0	1	1	1

c4		
e3	F	T
F	1	0
T	0	1

e1		
c1	F	T
F	1	.7
T	0	.3

e2		
c2	F	T
F	1	.7
T	0	.3

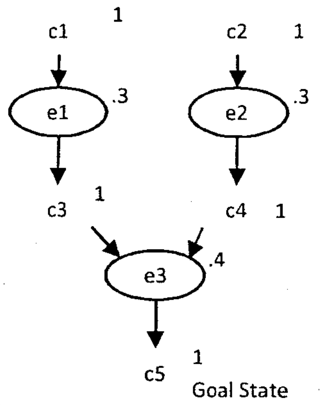
e3		
c3	F	T
F	1	.6
T	0	.4

Figure 6: Case 2

Clearly, the probabilistic score of case 2 is greater than that for case 1. This satisfies the intuitive property whereby a security metric should satisfy the concept that as more paths to a goal state exist, the security of the network decreases. Thus this simple example validates this notion and the use of the probabilistic score as a security metric.

Case 3: Both $e1$ and $e2$ must be exploited.

Figure 7 shows a case of the conjunctive relationship between exploits. The result in this case is $P(c5 = T) = 0.036$. That is, the probability of achieving the goal state is significantly less than in cases 1 and 2. This meets the intuition that it is now more difficult to exploit $e3$ compared to previous two cases.



c1	
F	0
T	1

c2	
F	0
T	1

c3		
e1	F	T
F	1	0
T	0	1

c4		
e2	F	T
F	1	0
T	0	1

c5		
e3	F	T
F	1	0
T	0	1

e1		
c1	F	T
F	1	.7
T	0	.3

e2		
c2	F	T
F	1	.7
T	0	.3

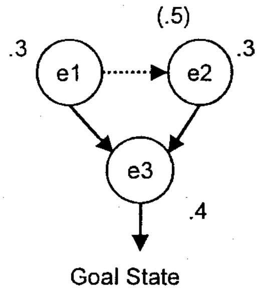
e3				
c3	F		T	
c4	F	T	F	T
F	1	1	1	.6
T	0	0	0	.4

Figure 7: Case 3

Case 4: A Successful exploitation affects other exploits with disjunctive relationships

In Case 2, exploits $e1$ and $e2$ are mutually independent, whereas in Case 4 shown in Figure 8, the dotted line indicates that the likelihood of exploit $e2$ depends on exploit $e1$. In particular, an attacker may have gained knowledge and skills following exploit $e1$, which would increase his chances in exploiting $e2$. In Case 4, the likelihood of successfully exploiting $e2$ without considering $e1$ is at 0.3 (same as in case 2), but this value will change to 0.5 if the effect of $e1$ is considered. This will result from the recomputed BS score using the adjusted scores for the AV, AC, Au, C,I and A parameters as described in section 4.1.2.

It is interesting to note that the probability of satisfying the goal condition is the same



e1	
T	F
.3	.7

	e2	
e1	T	F
T	.5	.5
F	.3	.7

		e3	
e1	e2	T	F
F	F	0	1
F	T	.4	.6
T	F	.4	.6
T	T	.4	.6

Figure 8: Case 4

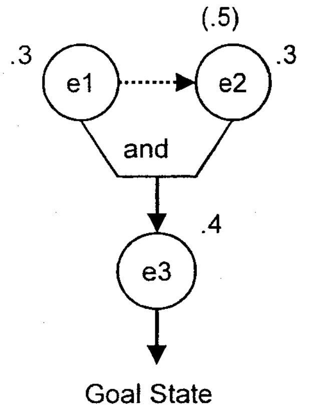
value 0.204 as in Case 2. An interpretation of this result is that in order to exploit *e*3 we must have either a successful exploitation of *e*1 or *e*2. If *e*1 is successfully exploited first, the likelihood of *e*2 increases. However, the attacker can go directly to *e*3 without attempting to exploit *e*2 (in which case the adjusted score makes no difference), which is the same as in case 2. If only *e*2 is exploited, then *e*1 can be ignored.

Case 5: A Successful exploitation affects other exploits with conjunctive relationships

The case shown in Figure 9 is similar to case 3 with the exception that *e*1 and *e*2 are dependant in the sense that successfully exploiting *e*1 increases the likelihood of exploiting *e*2. The result is 0.06 in this case. The calculation is simply based on the adjusted value of *e*2, since *e*1 must always be exploited, which in turn means that *e*2 will always take on the adjusted value.

4.3 Applying the Model

We now show how the model can be applied to handle some interesting cases. Consider the two AGs in Figure 10. In the graph on the left, an attacker must exploit A or B, then C and D in order to reach the goal state. The graph on the right differs slightly. In order to achieve the goal state, an attacker must execute the same steps as those to the left. However, if the attacker exploits A, he acquires knowledge that will make exploiting D easier. This



e1	
T	F
.3	.7

e2		
e1	T	F
T	.5	.5
F	.3	.7

e3			
e1	e2	T	F
F	F	0	1
F	T	0	1
T	F	0	1
T	T	.4	.6

Figure 9: Case 5

is denoted by the score of 0.4 (when A is not exploited before reaching D) and 0.8 (when A has been exploited before reaching D). The case can be modelled using the BN model as shown in Figures 11 and 12. The probability scores for reaching the goal states are 0.0816 and 0.1296, respectively, which matches the aforementioned intuition.

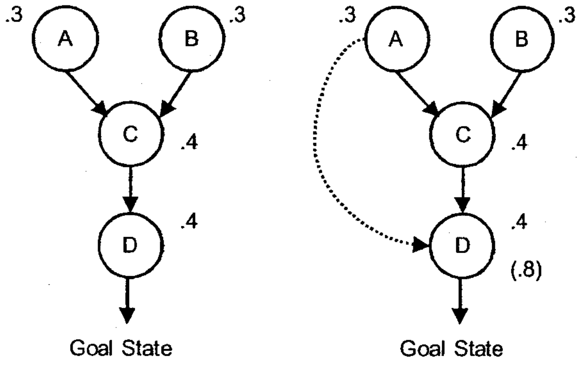


Figure 10: An Example of Applying the Model

Figure 13 shows another example of applying the model. Using the propagation of probabilities approach in Section 3.4, we would have $P(CVE - 2006 - 5302(1,2)) = P(user(1) * P(trust2, 1) * p(CVE - 2006 - 5302(1,2)) = .3433$. However, upon closer

A	
T	F
.3	.7

B	
T	F
.3	.7

C			
A	B	T	F
F	F	0	1
F	T	.4	.6
T	F	.4	.6
T	T	.4	.6

D		
C	T	F
F	0	1
T	.4	.6

Figure 11: Conditional Probability Tables for the Left Side Case

A	
T	F
.3	.7

B	
T	F
.3	.7

C			
A	B	T	F
F	F	0	1
F	T	.4	.6
T	F	.4	.6
T	T	.4	.6

D			
A	C	T	F
F	F	0	1
F	T	.4	.6
T	F	0	1
T	T	.8	.2

Figure 12: Conditional Probability Tables for the Right Side Case

scrutiny, this method of calculation is valid only if $trust(2,1)$ and $user(1)$ are mutually independent in which case $P(trust(2,1)|user(1)) = P(trust(2,1) = .5859$ leads to $P(CVE - 2006 - 5302(1,2)) = .3433$. However, this is clearly not the case. In fact, our model will show $P(trust(2,1)|user(1)) = .75$ which yields $P(CVE - 2006 - 5302(1,2)) = .4395$. Therefore, our model can deal with this case correctly while the previous method is not sufficient.

Although our BN-based model is more general and can handle cases where the probability propagation method in Section 3.4 cannot, the latter can be more efficient in its applicable cases. This is partly due to the fact that BNs constructed under our model have

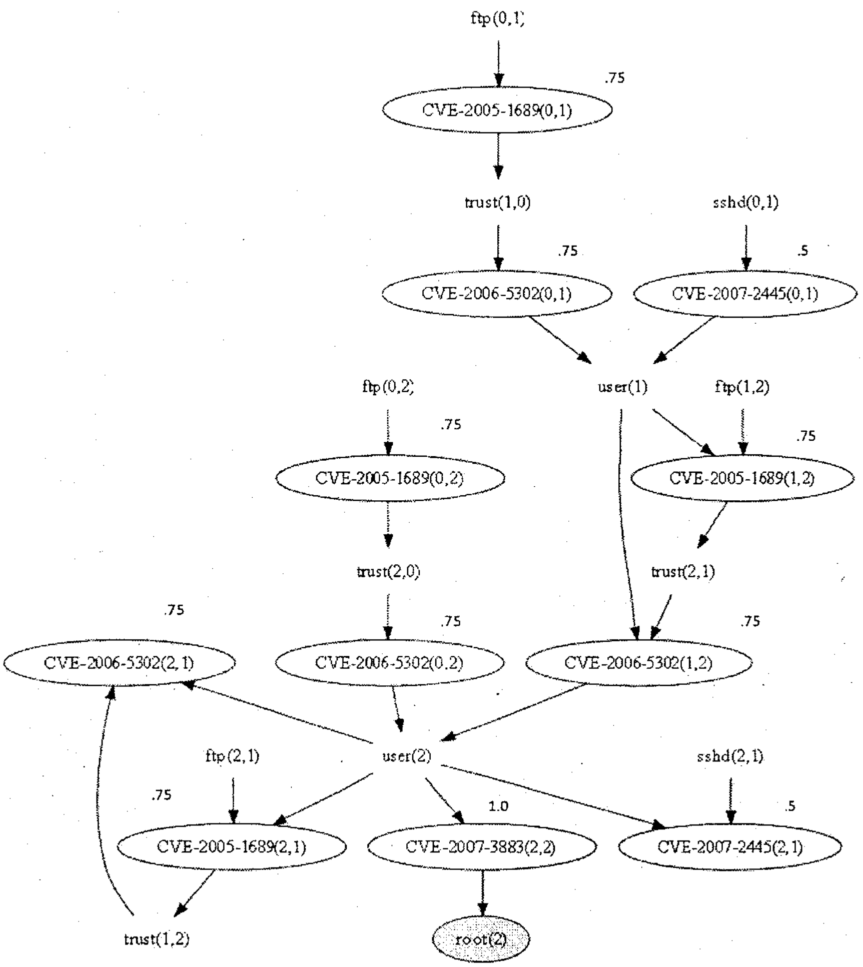


Figure 13: Another Example of Applying the Model

very special structures. More specifically, we use special conditional probabilities of 0 and 1 to model the conjunctive and disjunctive relationships between exploits, while most other conditional probabilities will not be useful in the modeling process. Therefore, a general-purpose BN inference may not be as efficient as the probability propagation method that is equivalent to a BN inference in special cases.

We thus propose a *Hybrid* model in which we combine the probability propagation method in Section 3.4 with our BN-based method. More specifically, we search backward from the goal condition in an AG to find subgraphs of the AG that are trees. We apply the probability propagation method to such subgraphs while adopting the BN-based method in other cases, as illustrated in Figure 14.

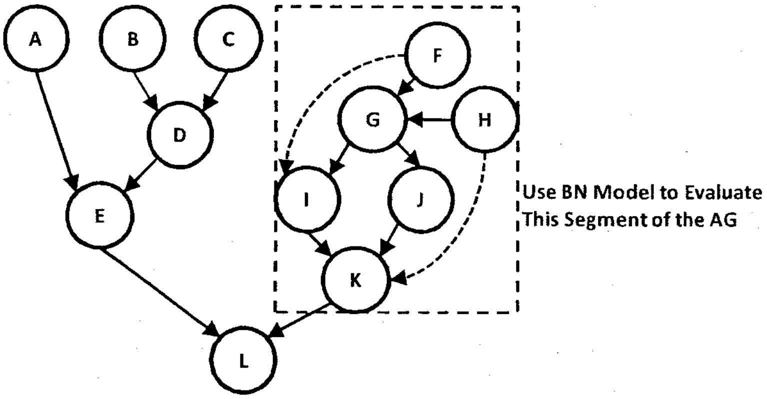


Figure 14: AG Hybrid Model

Chapter 5

Dynamic Bayesian Network-Based Attack Graph

5.1 The Model

As described in Section 3.2, CVSS provides several temporal metric scores in addition to base metric scores in order to model the time variant factors in determining the severity of a vulnerability. Such scores are, however, still intended for individual vulnerabilities and not for the overall security of a network. The objective of this research is to evolve the aforementioned BN-based model to a DBN model that can model the security of dynamically changing networks. The temporal links between time slices of the DBN will be established between the unobservable variables of the model. With these links, the model will enable the inference of the unobserved variables based on the observed variables within the same time slice and those of the previous slice of the DBN.

The model introduces three additional sets of vertices into the previous BN model. The first is the collection of E vertices that correspond to the *Exploitability* scores of the vulnerabilities. The second is the collection of RL vertices that correspond to the *Remediation*

Level scores. Finally, the third is the collection of *RC* vertices that correspond to the *Report Confidence* scores. These temporal metrics are defined in Section 3.2. The existing exploit vertices will then carry the final metric score -Temporal Score TS (instead of the BS in the static case), which has a similar role as the calculated scores in the case of the static domain as described in Chapter 4. However, in the static domain, the final score for an exploit is calculated based on its BS and the causal relationship between this exploit and others, whereas in the dynamic domain, the final score of each exploit will depend on three factors: the temporal score, the causal relationship between this exploit and others within the same time slice, and the causal relationships with exploits in the previous time slice (this will become clearer later with discussions using concrete cases).

Formally, given an attack graph G as a directed graph $G(E \cup C, R_r \cup R_i)$, we define E_E, E_{RL} and E_{RC} with the same cardinality as E to represent the set of E,RL and RC nodes. We then obtain an enriched set of nodes as $E' = E \cup E_E \cup E_{RL} \cup E_{RC}$. Let G' be the directed graph corresponding to E' in which the relations R_r and R_i remain the same. Then we can have the one slice BN as a pair (G', Q) where Q represents the conditional probabilities assigned as before. We then define a DBN as a pair (B_0, B_d), where B_0 defines the prior $P(X_1)$, and B_d is a two-slice temporal Bayes net(2TBN) that defines $P(X_t|X_{t-1})$ by means of a DAG: $P(X_t|X_{t-1}) = \prod_{i=1}^{N} P(X_t^i|parents(X_t^i))$.

For B_0, conditional probabilities are assigned in a similar way as in the static case except that now the model uses the TS scores instead of the BS scores. More specifically, the TS scores are derived as the product of BS and TGS using Equation 3. The derived TS scores are then assigned as conditional probabilities based on Equation 6. For B_d, the assignment of interslice conditional probabilities will depend on specific requirements of applications, since different variables in a time slice may be regarded as unobservable, and the effect of a previous slice will depend on the semantics of the variables in question. To make the discussions more concrete, this thesis shall discuss cases to illustrate the potential of the model.

The use of the DBN attack graph model offers the possibility of using a rich repertoire of DBN theory and advanced applications to allow researchers in the field of network security to solve network security problems such as, inferring node values, learning about unknown (hidden) parameters based on observed characteristics of the network, determining most probable explanations (MPE) for the networks (ie. most probable explanation that a particular set of nodes have been successfully exploited) and others.

5.2 Application 1: Inference of Exploit Node Values

In this proposed application, we derive the probability values for the exploit nodes (the TS scores) which represent the probability of successful exploitation. Recall, that the values of these exploit nodes represents the probability that an attacker will exploit this vulnerability. The capability of determining the TS score for each vulnerability is of interest (for example, to security administrators of a network) and needs to be derived from the base metric scores, temporal metric scores, intraslice and interslice dependencies. The model for this application is illustrated in Figure 15. In the model, the *observable* criteria are illustrated in the nodes with thin double lines and the *unobservable* criteria are illustrated in the nodes with the single thick line.

The objective of this application is to infer the values of the exploit nodes at different time intervals (time slices) based on observed criteria. In this case, a security administrator can observe the E, RL and RC metric values for each exploit of the graph at each time slice. Based on these observed values, the security administrator can use the model to infer the probability values for each of the exploit nodes. In addition, the application can be used to forecast future security characteristics of a network by integrating interesting work by Jonsson et al. [21] in which the attacking phases within a network are defined. We could quantify the security threat with respect to known attacking phases, or inversely, to

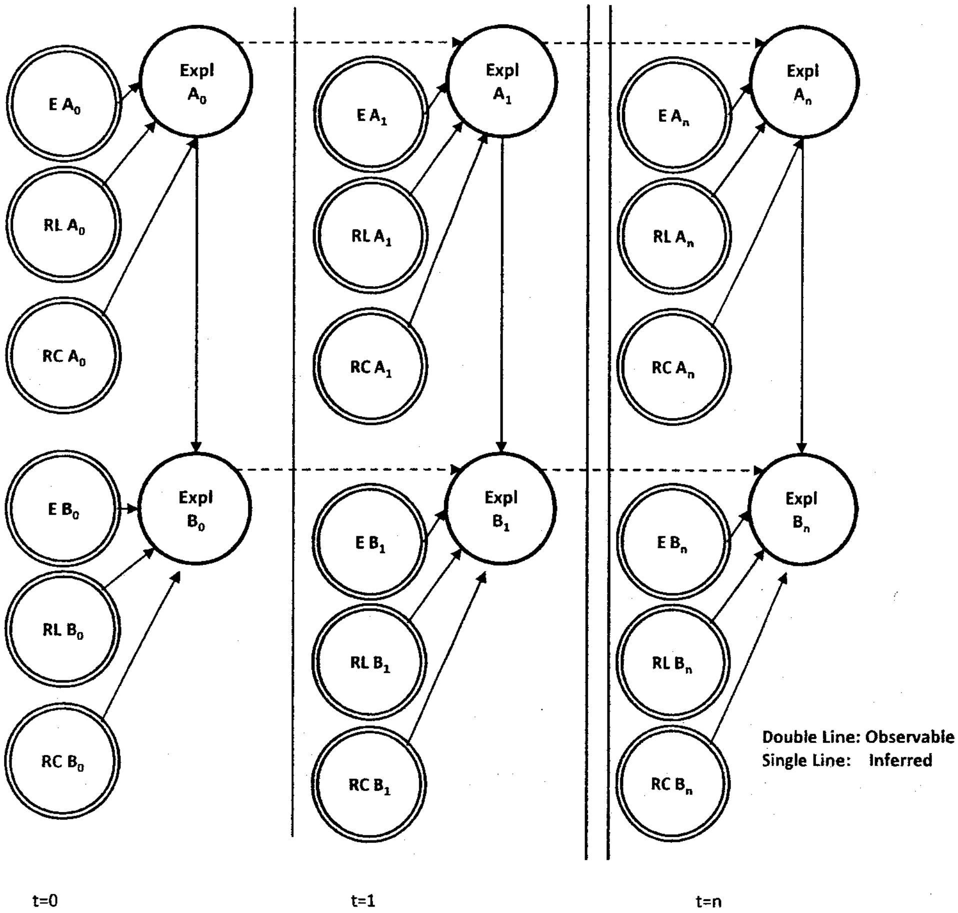


Figure 15: Inference of Exploit Node Values

infer the attacking phases based on the probability values computed from our model. Our application could also be used within the framework discussed in the introduction section to allow an administrator to establish a threshold value beyond which corrective actions must be taken. By plotting the node values as a function of the time slices, the administrator could make more effective decisions as to when corrective action should be taken.

Next we address the effect of a previous time slice on the present time slice as represented by the interslice arcs and the corresponding CPD. In the context of our model, the interslice dependencies will be application dependant and thus be user defined values. We introduce a variable τ, namely, the *Exploit Temporal Coefficient*. We use τ to adjust the

TS score of an exploit in a time slice (t) based on whether or not the same exploit was successfully exploited in the previous time slice ($t - 1$). In the case where an exploit has been successfully exploited in a previous time slice, τ will increase the TS score of the exploit for the present time slice. The possible value of τ ranges from 1, where the successful exploit of the vulnerability in a previous time slice has no effect on its TS score for the present time slice, to $1/(BS * TGS)$, which will result in a TS value of 1 meaning that once an exploit has been successfully exploited in a previous time slice, it will always be considered exploited in subsequent time slices.

This latter case implies that an attacker never relinquishes acquired knowledge or capabilities. This is reasonable when an attacker has compromised a vulnerability in a previous time slice (e.g., successfully uploading a malware that provides user accesses on a remote machine). In subsequent time slices, the attacker may retain this same remote access despite the possibility of having the previous vulnerability fully patched. In such a case, the interslice CPD assigning a TS value of 1 to an exploit node is appropriate. On the other hand, changes to the temporal component metrics may affect the overall network security in such a way that a previously possible exploit may become inaccessible (e.g., if the RL factor was *Temporary Fix* in the previous time slice and it now becomes *Official Fix*). Therefore, the user defined Exploit Temporal Coefficient (τ) will allow our model to handle different scenarios.

Definition 6 *Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$, we define a two-slice temporal Bayes net(2TBN) (B_0, B_d) based on a DAG in which arcs point from the set of observable nodes $O = \{E_E, E_{RL}, E_{RC}\}$ to the set of unobservable node $U = \{expl\}$. Given a temporal coefficient $\tau_{E_i} \in [1, 1/(BS_{E_i} * TGS_{E_i})]$, we assign inter-slide probabilities as $P(E^i_t = T \mid E^i_{t-1} = T) = \tau_E * BS_{E^i} * TGS_{E^i}$ and $P(E^i_t = T \mid E^i_{t-1} = F) = BS_{E^i} \cdot TGS_{E^i}$.*

5.2.1 An Example

To security administrators, the score of each exploit is unobservable, whereas the E, RL and RC scores are observable for each vulnerability by consulting resources such as the NVD [28]. To model the temporal dependencies between time slices, arcs linking the time slices are introduced between the exploit vertices since they are unobservable. Our objective is to infer their values and calculate the likelihood of attackers reaching the goal state. For this example, we consider only the E and RL temporal metrics for simplicity.

Figure 16 shows the DBN model in this case through a toy example of two exploits. In the model, the exploit vertices *addusrphp* and *sunvect* for this example are defined to be conditionally dependant on their respective E and RL vertex values as represented graphically in Figure 16. Note that it was decided not to include a node representing the Base Score in this version of the model since it is a fixed value and is invariant throughout the time slices. The model does, however, use the value of the Base Score as input into the calculation of the TS score as shown in Equation 3. In the example, the value of exploit *sunvect* is conditionally dependant on the value of exploit vertex *addusrphp*. This causal relationship implies that vulnerability *addusrphp* must be exploited first in order for vulnerability *sunvect* to be exploited. In this example, the goal state is the successful exploitation of vulnerability *sunvect*.

To model the temporal dependencies, arcs linking the time slices are introduced between *addusrphp* and *sunvect* (unobserved parameters). To complete the model, it is necessary to develop the CPDs for the intraslice relationships (within the same time slice) and the interslice relationships (from one time slice to the next).

We will illustrate this application with two options each using the same DBN (B_0, B_d) with the difference that the first sets $\tau_{php} = 1.81$ $(1/(BS_{php} * min(TGS_{php})))$ and $\tau_{sun} = 1.36$ $(1/(BS_{sun} * min(TGS_{sun})))$. The effect of this is that $P(E_t^i = T \mid E_{t-1}^i = T) = 1$ for all of the exploit nodes. This implies that once an attacker has successfully exploited a

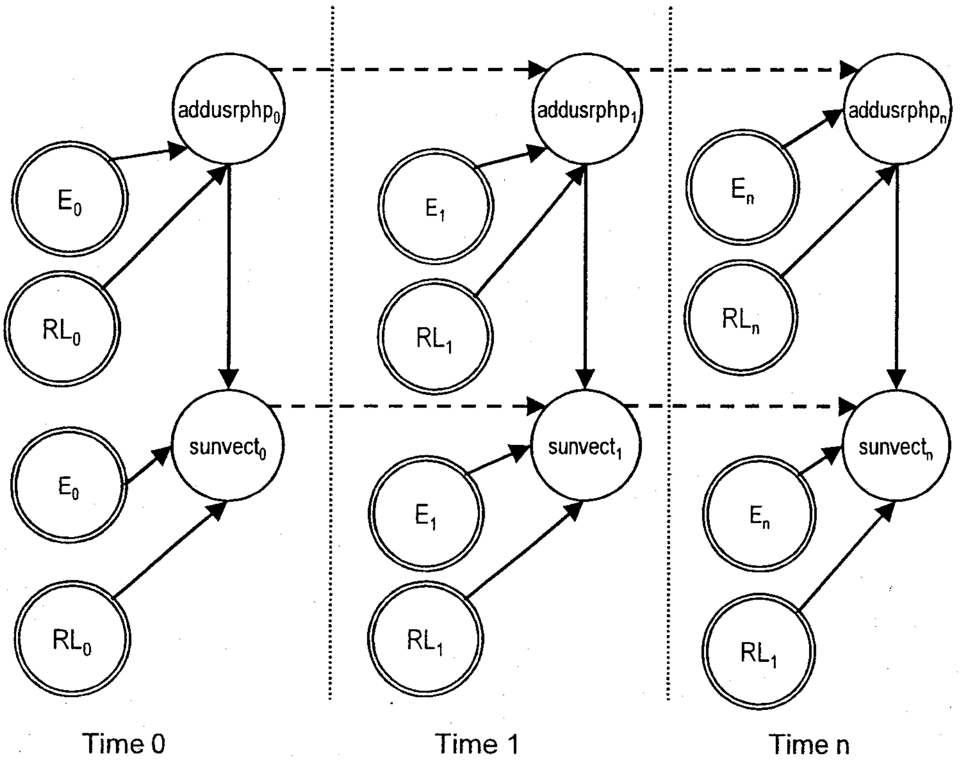


Figure 16: The DBN Model

node, he never relinquishes this ability and the model assigns the value for all subsequent time slices such that the node is considered exploited. Figures 17 and 18 show the intraslice CPDs in tabular format whereas Figures 19, 20 and 21 shows the interslice CPDs.

For the second option, we assign $\tau_{php} = 1.25$ and $\tau_{sun} = 1.1$. Although the model uses the same intraslice CPDs, the interslice CPDs will be modified and are shown in Figures 24 and 25. Notice that the table entries are calculated using CVSS equations as explained in section 3.2.

Suppose the objective is to calculate the probability value of an attacker successfully exploiting *sunvect* for any time slice. From NVD we obtain the BS for each vulnerability as follows: $BS(addusrphp) = 7.5$ and $BS(sunvect) = 10.0$. In this example, we consider only the E and RL temporal metrics from CVSS. Because they are observable for all time

	php$_0$		
RLphp$_0$	Ephp$_0$	F	T
OF	U	0.4454	0.5546
	POC	0.4127	0.5873
	F	0.3801	0.6199
	H	0.3475	0.6525
TF	U	0.4262	0.5738
	POC	0.3925	0.6075
	F	0.3587	0.6413
	H	0.325	0.675
W	U	0.3944	0.6056
	POC	0.3587	0.6413
	F	0.3231	0.6769
	H	0.2875	0.7125
U	U	0.3625	0.6375
	POC	0.325	0.6750
	F	0.2875	0.7125
	H	0.25	0.750

RLphp$_0$			
OF	TF	W	U
0.2	0.1	0.2	0.5

Ephp$_0$			
U	POC	F	H
0.7	0.1	0.1	0.1

Figure 17: Intraslice CPDs for php (time slice 0)

slices, we can set them to any values from the set E={U,POC,F,H} and RL={OF,TF,W,U} which correspond to the CVSS domain values for these metrics. To illustrate the model, we show the results for 3 example runs of the application. The values used for each of these runs is shown in Figure 23 for the first option and in Figure 27 for the second. The resulting value for the probability of exploitation of *sunvect* for each time slice is shown in Figure 22 and Figure 26.

sun$_0$									
php$_0$	RLsun$_0$	Esun$_0$	F	T	php$_0$	RLsun$_0$	Esun$_0$	F	T
F	OF	U	1	0	T	OF	U	0.2605	0.7395
		POC	1	0			POC	0.217	0.783
		F	1	0			F	0.1735	0.8265
		H	1	0			H	0.13	0.87
	TF	U	1	0		TF	U	0.235	0.765
		POC	1	0			POC	0.19	0.81
		F	1	0			F	0.145	0.855
		H	1	0			H	0.10	0.90
	W	U	1	0		W	U	0.1925	0.8075
		POC	1	0			POC	0.145	0.855
		F	1	0			F	0.0975	0.9025
		H	1	0			H	0.05	0.95
	U	U	1	0		U	U	0.15	0.85
		POC	1	0			POC	0.10	0.90
		F	1	0			F	0.05	0.95
		H	1	0			H	0	1

Esun$_0$			
U	POC	F	H
0.7	0.1	0.1	0.1

RLsun$_0$			
OF	TF	W	U
0.2	0.1	0.2	0.5

Figure 18: Intraslice CPDs for sunvect

5.3 Application 2: Inference of TGS Node Values

Now we consider applications where the temporal metric scores of a vulnerability are of interest (for example, to security vendors who maintain these scores) and can be derived from base scores and the observed TS or exploit node scores (determined from reported security incidents involving that vulnerability).

More formally, in this case, the DBN (B_0, B_d) defines a DAG including the same intraslice arcs as in Case Study 1, however, the interslice arcs now link some or all of the

Php$_i$ i≥1									
Php$_{i-1}$	RLphp$_i$	Ephpi	F	T	Php$_{i-1}$	RLphp$_i$	Ephpi	F	T
F	OF	U	0.4454	0.5546	T	OF	U	0	1
		POC	0.4127	0.5873			POC	0	1
		F	0.3801	0.6199			F	0	1
		H	0.3475	0.6525			H	0	1
	TF	U	0.4262	0.5738		TF	U	0	1
		POC	0.3925	0.6075			POC	0	1
		F	0.3587	0.6413			F	0	1
		H	0.325	0.675			H	0	1
	W	U	0.3944	0.6056		W	U	0	1
		POC	0.3587	0.6413			POC	0	1
		F	0.3231	0.6769			F	0	1
		H	0.2875	0.7125			H	0	1
	U	U	0.3625	0.6375		U	U	0	1
		POC	0.325	0.6750			POC	0	1
		F	0.2875	0.7125			F	0	1
		H	0.25	0.750			H	0	1

Figure 19: Interslice CPDs for PHP $\tau_{php} = 1/(BS_{php} * TGS_{php})$

temporal metric nodes (ie. E or RL) depending on the objective and upon which nodes are unobservable. The model for this application is illustrated in Figure 28. The exploit nodes are observed at regular time intervals or time slices. The intraslice relationships remain unchanged and only the interslice arcs change.

We shall infer only the E metric, although the model itself can also be used to infer other temporal metric node scores. Also, we shall rely on a simple choice regarding the transition probabilities of the E metric value from one time slice to the next. In this application, the exploit nodes are observable. That is, for each time slice, the value of each exploit node is observed to be either {T,F} (exploited or not). The values of the E metric can be inferred which can then be matched to a value in $\{U, POC, F, H\}$.

Sun_i i≥1											
php_i	Sun_{i-1}	$RLsun_i$	$Esun_i$	F	T	php_i	Sun_{i-1}	$RLsun_i$	$Esun_i$	F	T
F	F	OF	U	1	0	T	F	OF	U	0.2605	0.7395
			POC	1	0				POC	0.217	0.783
			F	1	0				F	0.1735	0.8265
			H	1	0				H	0.13	0.87
		TF	U	1	0			TF	U	0.235	0.765
			POC	1	0				POC	0.19	0.81
			F	1	0				F	0.145	0.855
			H	1	0				H	0.10	0.90
		W	U	1	0			W	U	0.1925	0.8075
			POC	1	0				POC	0.145	0.855
			F	1	0				F	0.0975	0.9025
			H	1	0				H	0.05	0.95
		U	U	1	0			U	U	0.15	0.85
			POC	1	0				POC	0.10	0.90
			F	1	0				F	0.05	0.95
			H	1	0				H	0	1

Figure 20: Interslice CPDs for SunVect Part 1 $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$

Definition 7 *Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$, we define a two-slice temporal Bayes net(2TBN) (B_0, B_d) based on a DAG in which the set of observable nodes $O = \{E\}$ and the set of unobservable node $U = \{E_E, E_{RL}, E_{RC}\}$.*

The inter-slice probabilities $P(E^i_{E_t} \mid E^i_{E_{t-1}}) = \sigma$ are to be assigned using user-defined values. In our case, we shall define σ in Figure 29.

5.3.1 An Example

To vendors who create and maintain the CVSS databases, temporal scores may be unobservable and must be estimated from base scores and reported security incidents. We now consider the case where the E temporal metrics for each vulnerability are unobservable. In the previous case, the model was able to observe the E metric values and then infer the Exploit node values. In this case, we have the reverse situation. The goal in this case is

Sun_i i≥1											
php_i	Sun_{i-1}	$RLsun_i$	$Esun_i$	F	T	php_i	Sun_{i-1}	$RLsun_i$	$Esun_i$	F	T
F	T	OF	U	1	0	T	T	OF	U	0	1
			POC	1	0				POC	0	1
			F	1	0				F	0	1
			H	1	0				H	0	1
		TF	U	1	0			TF	U	0	1
			POC	1	0				POC	0	1
			F	1	0				F	0	1
			H	1	0				H	0	1
		W	U	1	0			W	U	0	1
			POC	1	0				POC	0	1
			F	1	0				F	0	1
			H	1	0				H	0	1
		U	U	1	0			U	U	0	1
			POC	1	0				POC	0	1
			F	1	0				F	0	1
			H	1	0				H	0	1

Figure 21: Interslice CPDs for SunVect Part 2 $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$

to update the E Temporal Metric values for maintaining the CVSS databases based on the DBN model.

Figure 30 illustrates the DBN model for this case where only the unobservable E metric vertices are linked from one time slice to the next. The interpretation is that the value of the E metric in the previous time slice will have an impact on determining the likelihood of which state the E metric vertex will be in during the subsequent time slices. Figures 17 and 18 show the intraslice CPDs in tabular format whereas Figure 29 shows the interslice CPDs.

Suppose reported security incidents show that that *addusrphp* and *sunvect* have been observed to have the values indicated in Figure 31 for 5 time slices. The DBN model can then infer the probabilistic scores for each of the E nodes. For example, in time slice 2, the model infers that $P(Esun_2 = U) = 0.612$ whereas $P(Esun_2 = H) = 0.062$ implying

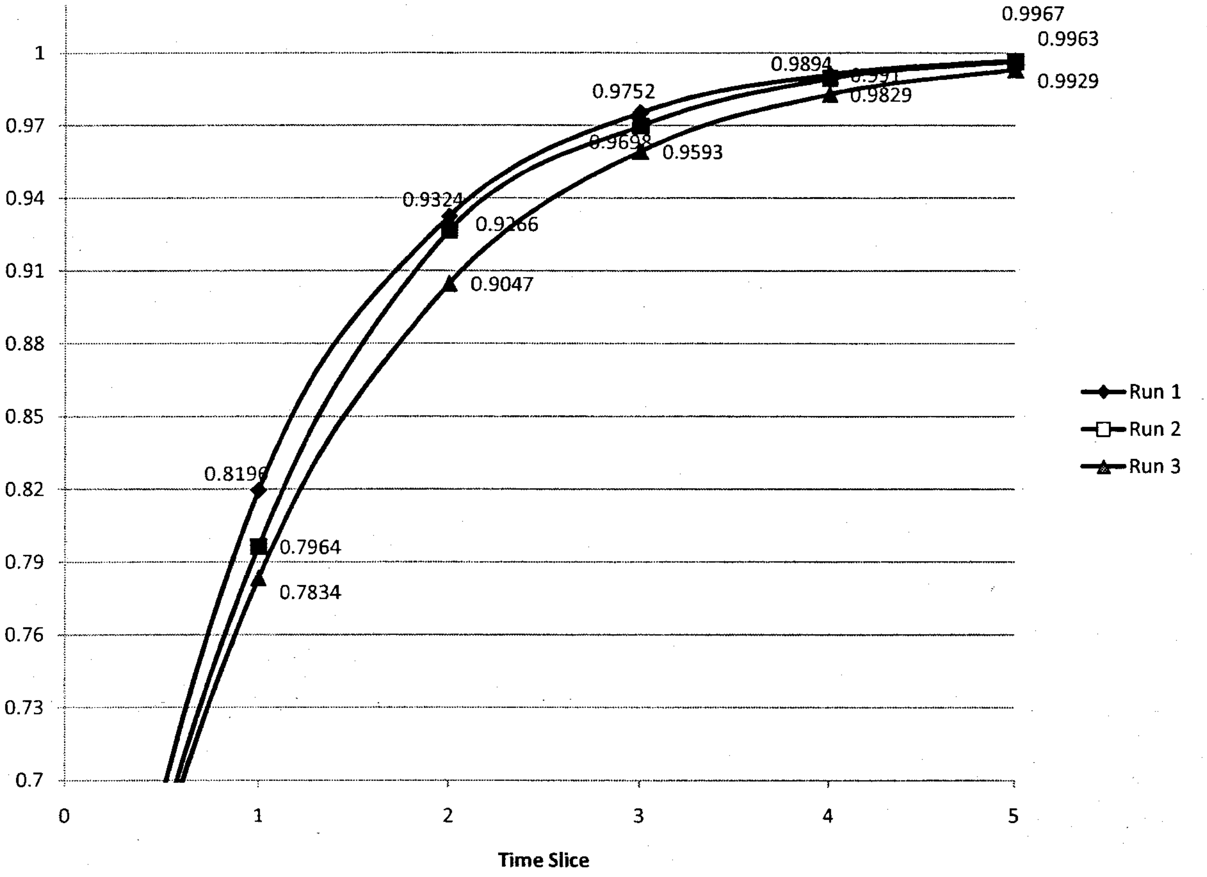
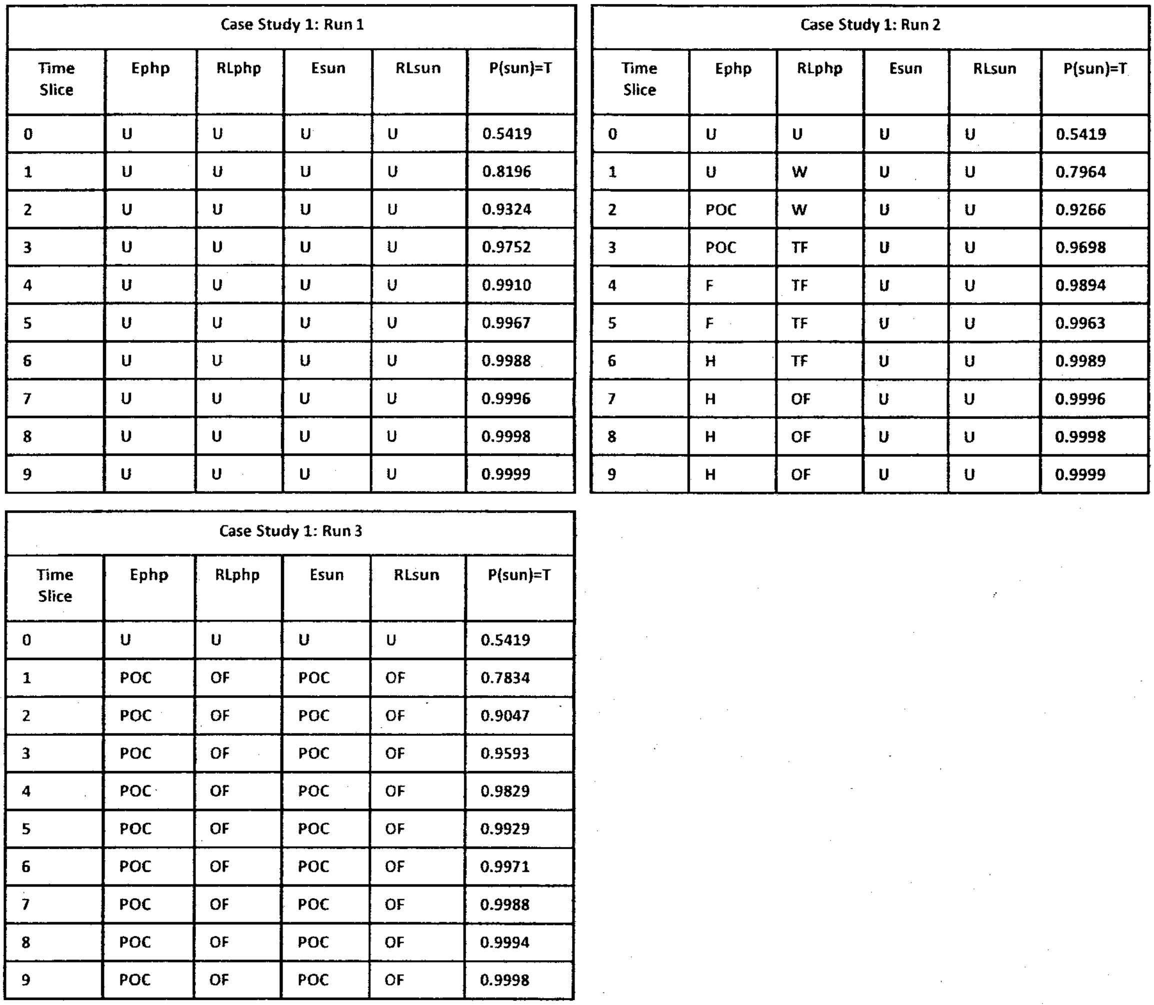


Figure 22: $P(sunvect) = T$ at Each Time Slice $\tau_{php} = 1/(BS_{php} * TGS_{php})$ and $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$

that it is ten times more likely that $Esun_3$ is in state U (Unproven) than in state H (High).

5.4 Case Study

Now we apply the DBN model to a more complex case to demonstrate its potential by examining the network from Figure 2. For this case, we will apply the methodology described in Application 1 of the model. That is, we assume E and RL are observable and the objective is to infer the values of the exploit nodes. Specifically, the example will examine the value of the goal state *local_bof(2)*.



Case Study 1: Run 1					
Time Slice	Ephp	RLphp	Esun	RLsun	P(sun)=T
0	U	U	U	U	0.5419
1	U	U	U	U	0.8196
2	U	U	U	U	0.9324
3	U	U	U	U	0.9752
4	U	U	U	U	0.9910
5	U	U	U	U	0.9967
6	U	U	U	U	0.9988
7	U	U	U	U	0.9996
8	U	U	U	U	0.9998
9	U	U	U	U	0.9999

Case Study 1: Run 2					
Time Slice	Ephp	RLphp	Esun	RLsun	P(sun)=T
0	U	U	U	U	0.5419
1	U	W	U	U	0.7964
2	POC	W	U	U	0.9266
3	POC	TF	U	U	0.9698
4	F	TF	U	U	0.9894
5	F	TF	U	U	0.9963
6	H	TF	U	U	0.9989
7	H	OF	U	U	0.9996
8	H	OF	U	U	0.9998
9	H	OF	U	U	0.9999

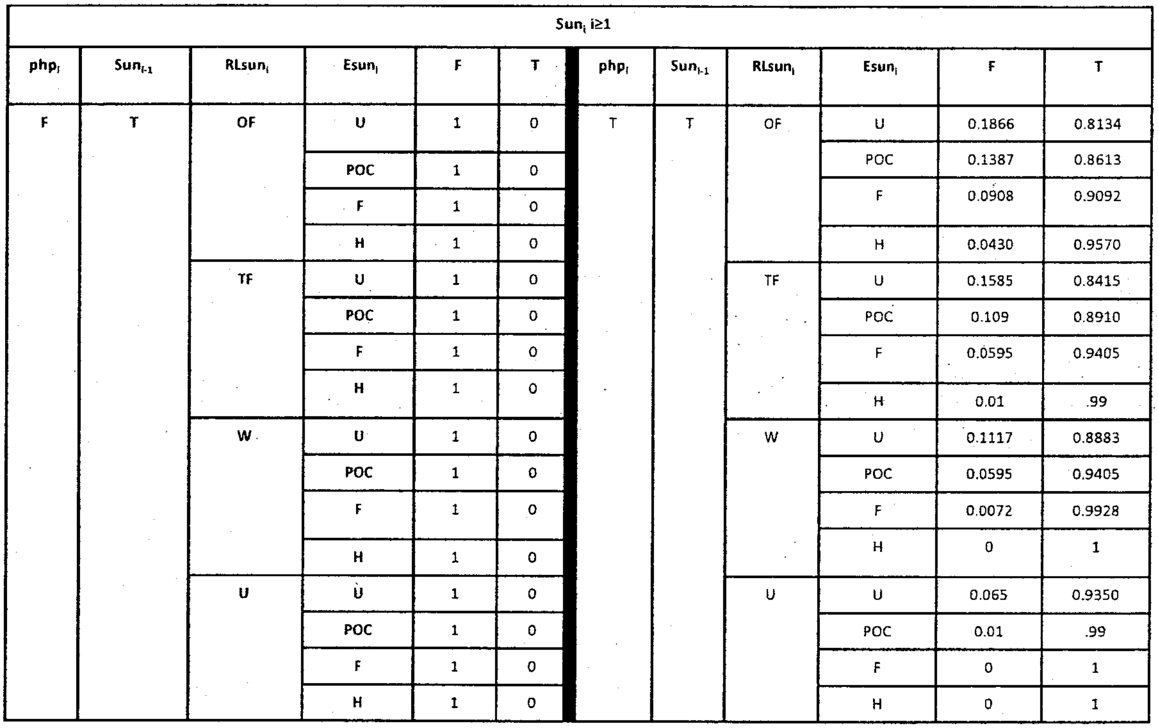
Case Study 1: Run 3					
Time Slice	Ephp	RLphp	Esun	RLsun	P(sun)=T
0	U	U	U	U	0.5419
1	POC	OF	POC	OF	0.7834
2	POC	OF	POC	OF	0.9047
3	POC	OF	POC	OF	0.9593
4	POC	OF	POC	OF	0.9829
5	POC	OF	POC	OF	0.9929
6	POC	OF	POC	OF	0.9971
7	POC	OF	POC	OF	0.9988
8	POC	OF	POC	OF	0.9994
9	POC	OF	POC	OF	0.9998

Figure 23: 3 Sample Runs $\tau_{php} = 1/(BS_{php} * TGS_{php})$ and $\tau_{sun} = 1/(BS_{sun} * TGS_{sun})$

Figure 32 shows the corresponding DBN produced by GeNIe [10]. Figure 33 shows 2 slices *unrolled* of the corresponding DBN network. We will run our DBN for 10 time slices and show the changing values for $P(local\_bof(2)) = T$. The example uses the following values as the Base Metric Score for each vulnerability: ftp_rhosts=0.8, rsh=0.9, sshd_bof=0.1 and local_bof=0.1. Figure 34 shows the results when E in all time slices is set to U and RL to TF.

Php$_i$ i≥1									
Php$_{i-1}$	RLphp$_i$	Ephpi	F	T	Php$_{i-1}$	RLphp$_i$	Ephpi	F	T
F	OF	U	0.4454	0.5546	T	OF	U	0	0.6933
		POC	0.4127	0.5873			POC	0	0.7341
		F	0.3801	0.6199			F	0	0.7749
		H	0.3475	0.6525			H	0	0.8156
	TF	U	0.4262	0.5738		TF	U	0	0.7173
		POC	0.3925	0.6075			POC	0	0.7594
		F	0.3587	0.6413			F	0	0.8016
		H	0.325	0.675			H	0	0.8438
	W	U	0.3944	0.6056		W	U	0	0.757
		POC	0.3587	0.6413			POC	0	0.8016
		F	0.3231	0.6769			F	0	0.8461
		H	0.2875	0.7125			H	0	0.8906
	U	U	0.3625	0.6375		U	U	0	0.7969
		POC	0.325	0.6750			POC	0	0.8438
		F	0.2875	0.7125			F	0	0.8906
		H	0.25	0.750			H	0	0.9375

Figure 24: Interslice CPDs for PHP $\tau_{php} = 1.25$



Sun_i i≥1											
php_i	Sun_{i-1}	$RLsun_i$	$Esun_i$	F	T	php_i	Sun_{i-1}	$RLsun_i$	$Esun_i$	F	T
F	T	OF	U	1	0	T	T	OF	U	0.1866	0.8134
			POC	1	0				POC	0.1387	0.8613
			F	1	0				F	0.0908	0.9092
			H	1	0				H	0.0430	0.9570
		TF	U	1	0			TF	U	0.1585	0.8415
			POC	1	0				POC	0.109	0.8910
			F	1	0				F	0.0595	0.9405
			H	1	0				H	0.01	.99
		W	U	1	0			W	U	0.1117	0.8883
			POC	1	0				POC	0.0595	0.9405
			F	1	0				F	0.0072	0.9928
			H	1	0				H	0	1
		U	U	1	0			U	U	0.065	0.9350
			POC	1	0				POC	0.01	.99
			F	1	0				F	0	1
			H	1	0				H	0	1

Figure 25: Interslice CPDs for Sun Vect $\tau_{sunvect} = 1.1$

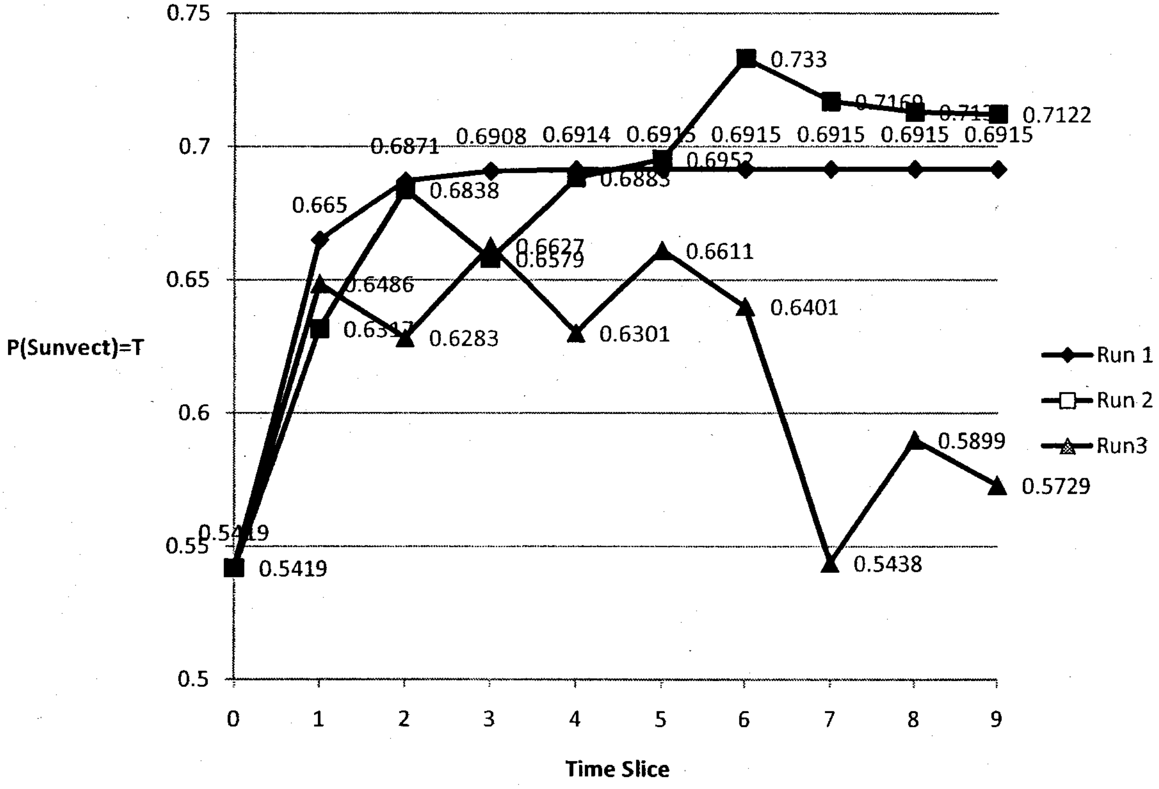
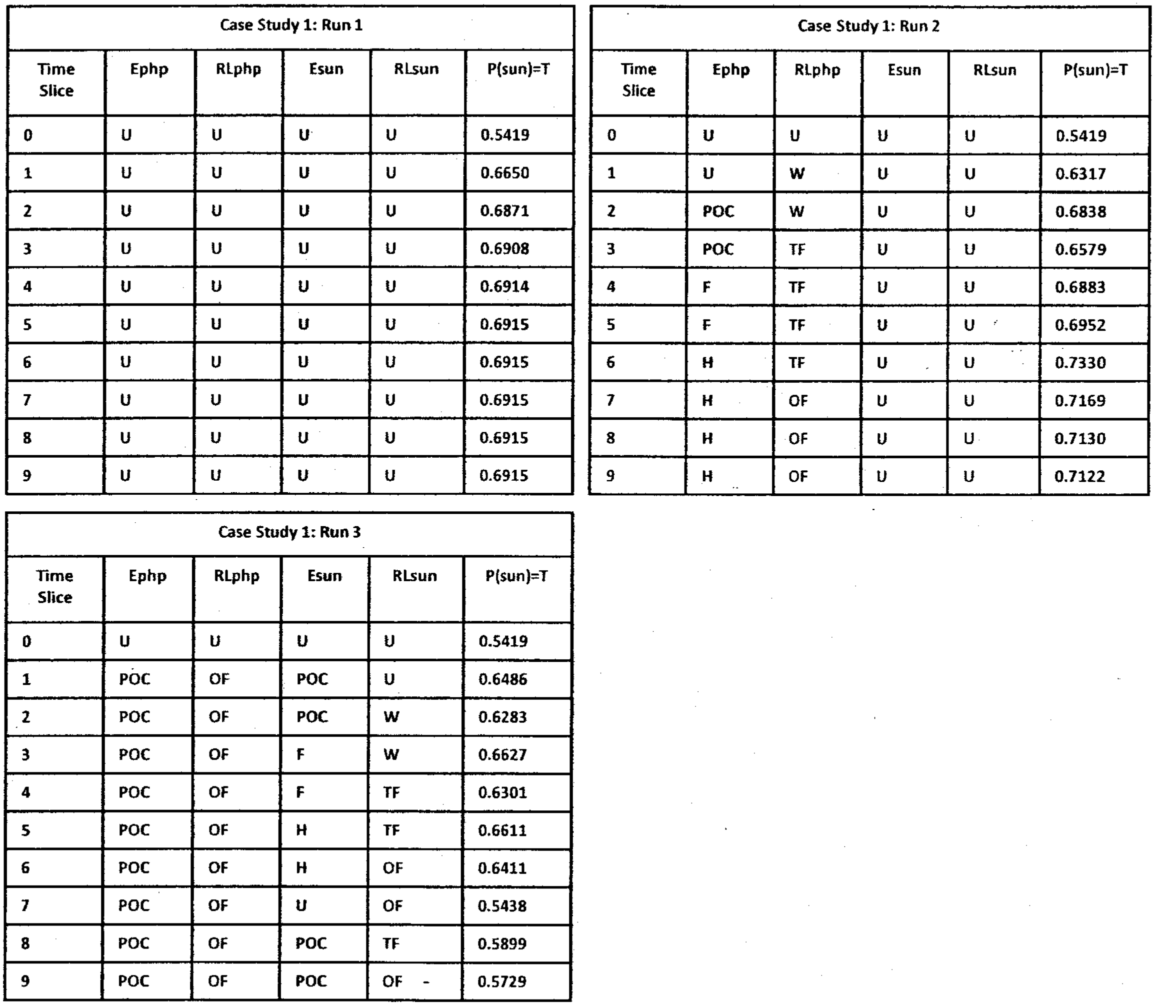


Figure 26: $P(sunvect) = T$ at Each Time Slice $\tau_{php} = 1.25$ and $\tau_{sun} = 1.1$



Case Study 1: Run 1					
Time Slice	Ephp	RLphp	Esun	RLsun	P(sun)=T
0	U	U	U	U	0.5419
1	U	U	U	U	0.6650
2	U	U	U	U	0.6871
3	U	U	U	U	0.6908
4	U	U	U	U	0.6914
5	U	U	U	U	0.6915
6	U	U	U	U	0.6915
7	U	U	U	U	0.6915
8	U	U	U	U	0.6915
9	U	U	U	U	0.6915

Case Study 1: Run 2					
Time Slice	Ephp	RLphp	Esun	RLsun	P(sun)=T
0	U	U	U	U	0.5419
1	U	W	U	U	0.6317
2	POC	W	U	U	0.6838
3	POC	TF	U	U	0.6579
4	F	TF	U	U	0.6883
5	F	TF	U	U	0.6952
6	H	TF	U	U	0.7330
7	H	OF	U	U	0.7169
8	H	OF	U	U	0.7130
9	H	OF	U	U	0.7122

Case Study 1: Run 3					
Time Slice	Ephp	RLphp	Esun	RLsun	P(sun)=T
0	U	U	U	U	0.5419
1	POC	OF	POC	U	0.6486
2	POC	OF	POC	W	0.6283
3	POC	OF	F	W	0.6627
4	POC	OF	F	TF	0.6301
5	POC	OF	H	TF	0.6611
6	POC	OF	H	OF	0.6411
7	POC	OF	U	OF	0.5438
8	POC	OF	POC	TF	0.5899
9	POC	OF	POC	OF	0.5729

Figure 27: 3 Sample Runs $\tau_{php} = 1.25$ and $\tau_{sun} = 1.1$

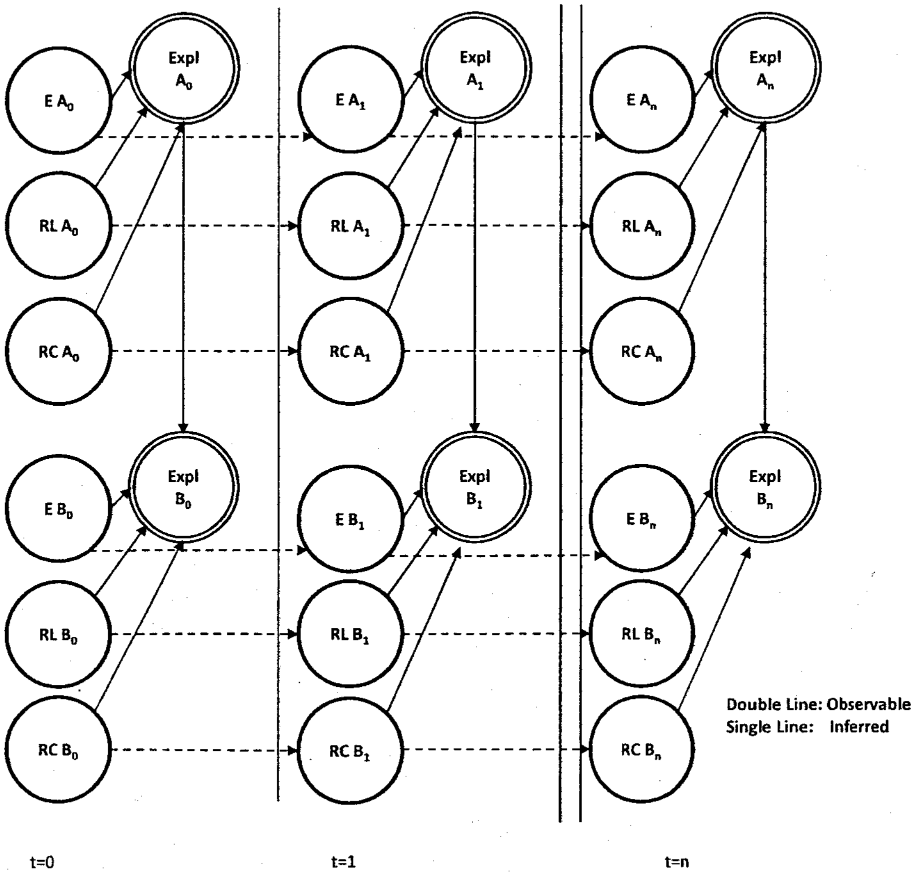


Figure 28: Inference of TGS Node Values

Ephp$_i$ i≥1				
Ephp$_{i-1}$	U	POC	F	H
U	.8	.1	.1	0
POC	0	0.8	0.1	0.1
F	0	0	0.9	0.1
H	0	0	0	1

Esun$_i$ i≥1				
Esun$_{i-1}$	U	POC	F	H
U	.8	.1	.1	0
POC	0	0.8	0.1	0.1
F	0	0	0.9	0.1
H	0	0	0	1

Figure 29: Interslice CPDs

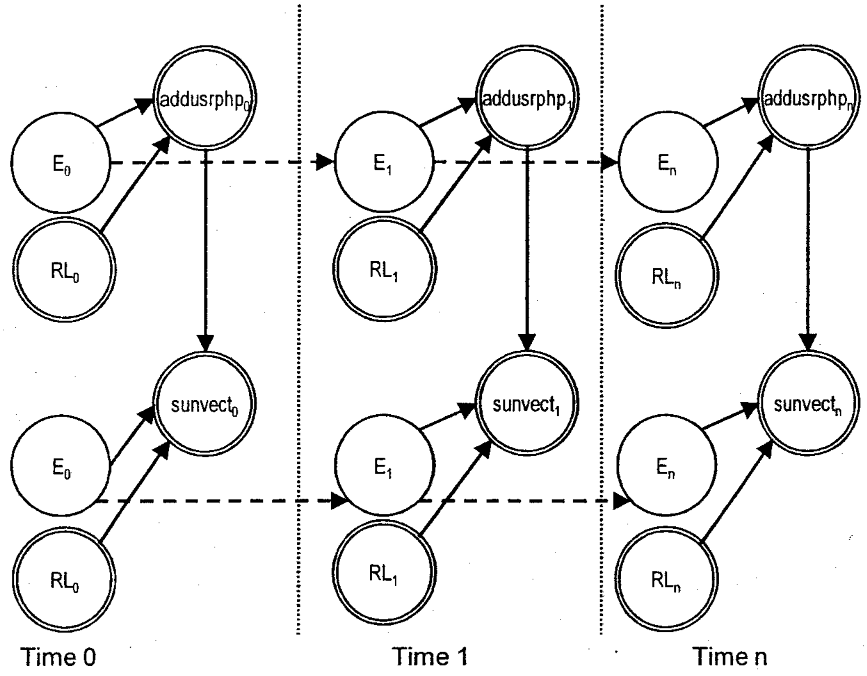
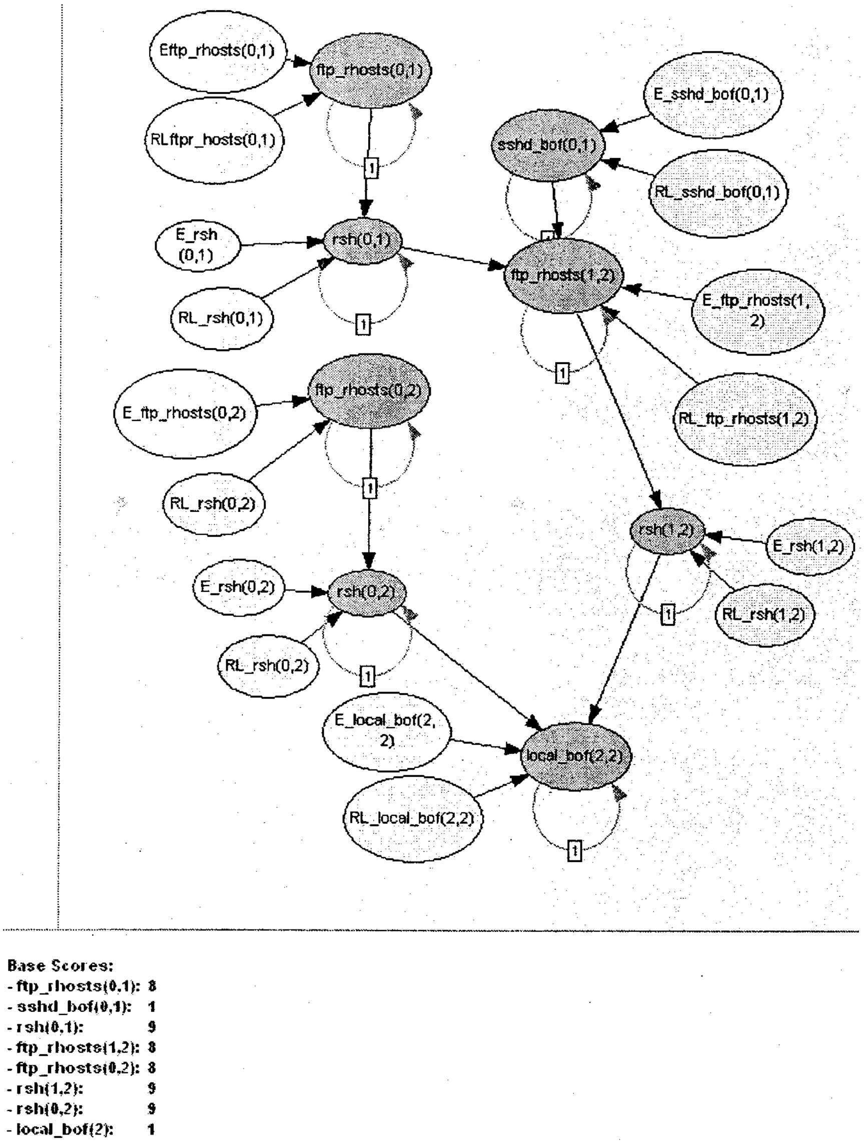


Figure 30: The DBN Model

Time Slice	php_i	$RLphp_i$	sun_i	$RLsun_i$	$Ephp_i$				$Esun_i$			
					U	POC	F	H	U	POC	F	H
0	F*	U*	F*	U*	0.736	0.099	0.088	0.077	0.814	0.083	0.063	0.040
1	F*	W*	F*	W*	0.581	0.157	0.166	0.096	0.705	0.135	0.111	0.049
2	T*	W*	F*	W*	0.441	0.185	0.239	0.134	0.612	0.174	0.152	0.062
3	T*	TF*	F*	TF*	0.341	0.192	0.285	0.183	0.509	0.203	0.202	0.086
4	T*	OF*	T*	OF*	0.268	0.186	0.312	0.234	0.40	0.212	0.258	0.130

***:Denotes Observed Value**

Figure 31: Inferred Values of E Metric When Exploit and RL Nodes Observed



```
Base Scores:
- ftp_rhosts(0,1): 8
- sshd_bof(0,1):   1
- rsh(0,1):        9
- ftp_rhosts(1,2): 8
- ftp_rhosts(0,2): 8
- rsh(1,2):        9
- rsh(0,2):        9
- local_bof(2):    1
```

Figure 32: DBN for a More Complex AG

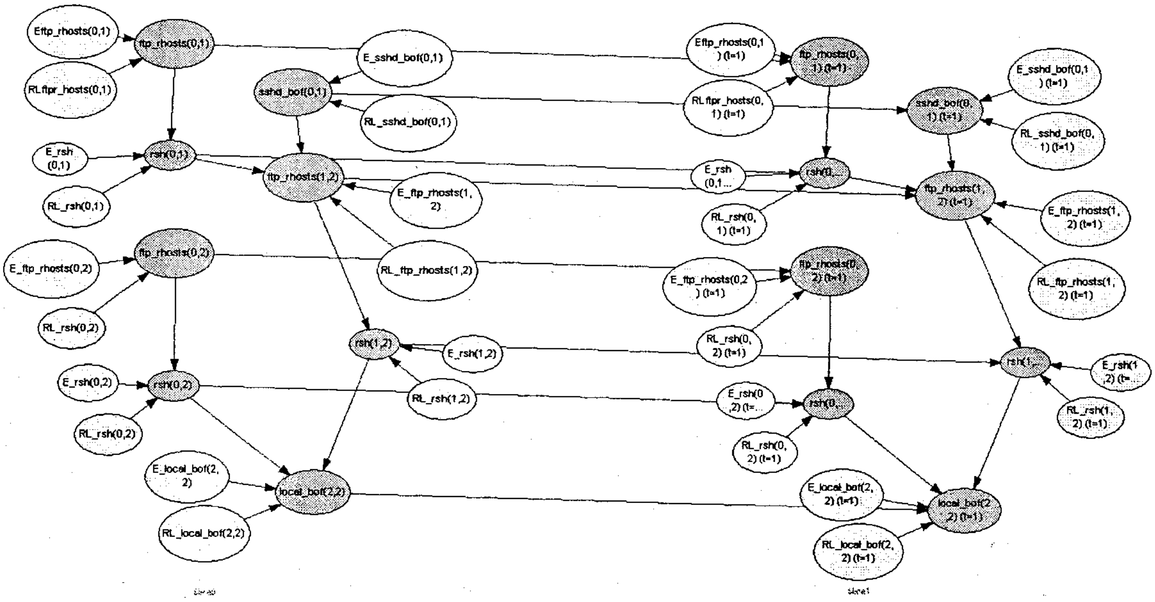


Figure 33: Two Slices

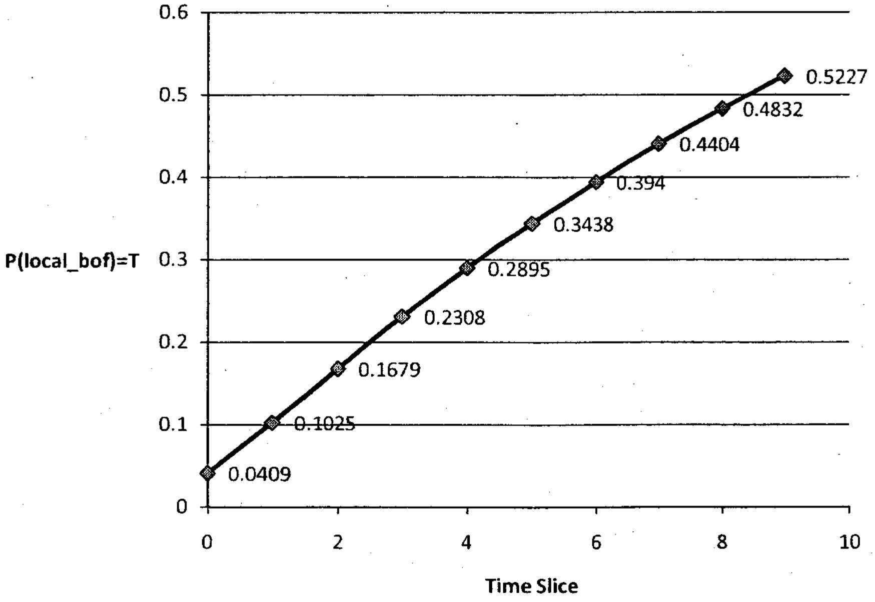


Figure 34: P(local_bof)=T for Each Time slice

Chapter 6

Implementation and Simulation

This section first describes the implementation of tools supporting the proposed models, and then studies the scalability of the models through simulation studies. The simulations were performed on a PC equipped with one Intel Core 2 Duo 2 GHz CPU, 4 GB of RAM, Microsoft Windows Vista (64 bits). For graph rendering we use the GraphViz visualization package [22]. For development, Netbeans 5.5 and jdk1.6.0_01 are used. Simulation studies were conducted in lieu of experiments due to the lack of availability of publicly available data sets of real world attack graphs.

6.1 Simulation Environment

In order to examine the scalability of the proposed models, several tools have been implemented or used. These tools together form the simulation environment in which our studies were conducted. We describe their purposes and details in this section.

Attack Graph Simulator The *Random Attack Graph Simulator* is a simulator implemented in [44] consisting of a set of Java programs that can generate random attack graphs based on given parameters, including the number of total AG nodes and the distribution of various dependency relationships, such as the ratio of initial conditions among all conditions. These parameters can be selected in order to generate a vast range of AG with varying characteristics to simulate attack graphs of different networks. As depicted in Figure 35, the simulator takes as inputs the total number of exploit nodes to be generated, the number of condition nodes in the attack graph, and the distribution of conditional dependency relationships. The simulator then generates an attack graph in the .dot format, which is the language used by the GraphViz visualization package [22].

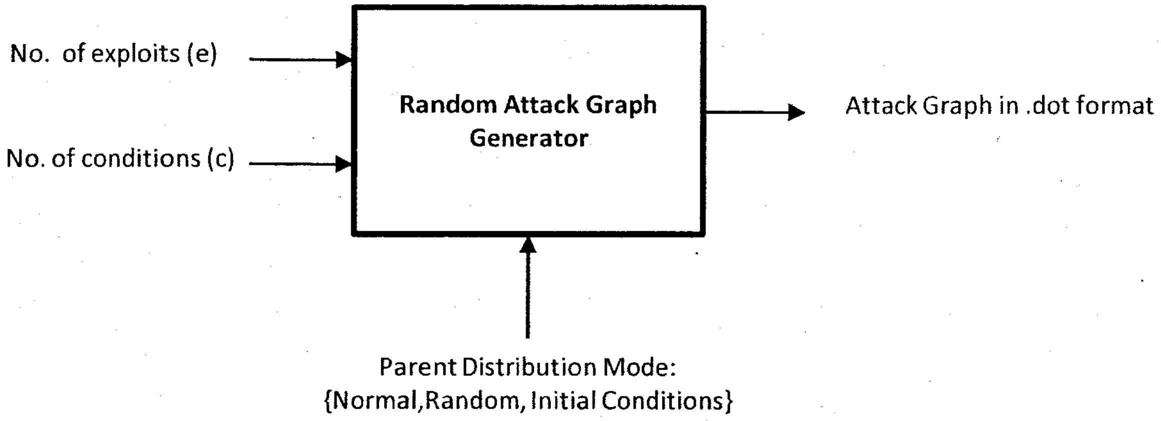

No. of exploits (e)
No. of conditions (c)
Random Attack Graph Generator
Attack Graph in .dot format
Parent Distribution Mode:
{Normal,Random, Initial Conditions}

Figure 35: Attack Graph Generator

Security Metric Calculator (Polaris) Developed for this thesis, the *Security Metric Calculator* named Polaris is a software program written in Java for applying security metrics to given attack graphs. From the .dot attack graph files loaded into the program, this application generates the annotated graphs by retrieving the CVSS scores from the NVD files also loaded into the program and assigning a probabilistic score to each node. The program has two modes as shown in Figure 36. In the *probability propagation* mode, the program will compute the overall probability of achieving the goal state using the probability propagation method introduced in 3.4. In the *BN* mode, the simulator will compute the BN model of the given attack graph and generate the CPTs for all nodes of the AG and the .xdsl file, which can subsequently be loaded into the GeNIe software program [10] for visualizing the BN model and performing inference.

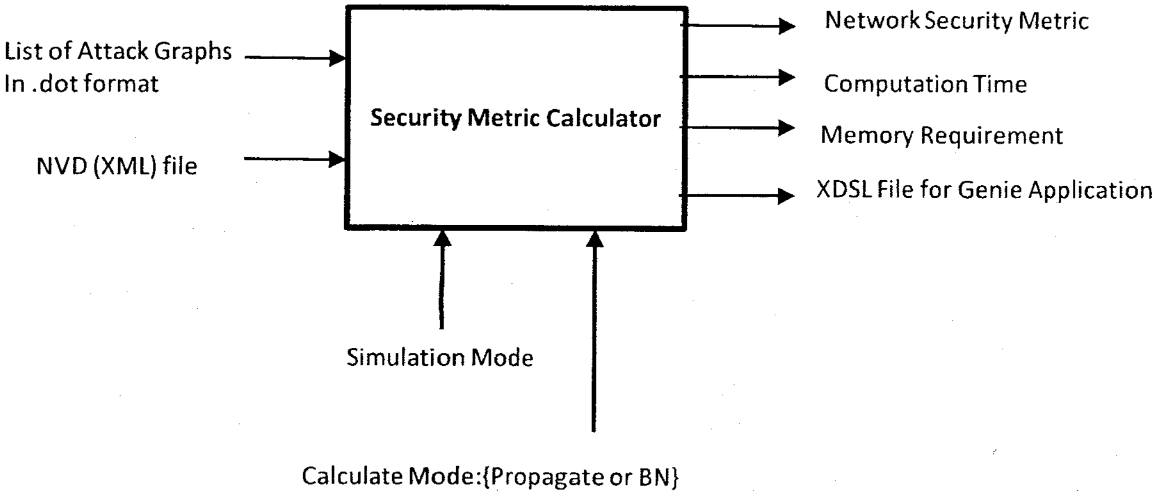


Figure 36: Simulator - Security Metric Calculator (Polaris)

BN Attack Graph Analyzer (Sirius) Sirius is a second software application developed for this thesis. As shown in Figure 37 it takes as input a specified list of BN attack graph models generated by Polaris (.xdsl files), and processes them in a batch in order to generate useful statistics in a comma separated value text file. This file can then be analyzed by any numerical analysis tool (such as Microsoft Excel). For the purpose of this thesis, the output allows us to study the variations in several characteristics of the proposed BN Attack Graph models. In particular, we will show later in this thesis, simulation results of the growth of BN attack graph model sizes with regards to the size of attack graphs, which are produced by Sirius.

Graphical Network Interface (GeNIe) GeNIe is a development environment for building graphical decision-theoretic models [10] developed at the Decision Systems Laboratory, University of Pittsburgh. GeNIe is implemented in Visual C++ based on the MFC (Microsoft Foundation Classes). GeNIe allows for building BN models of any

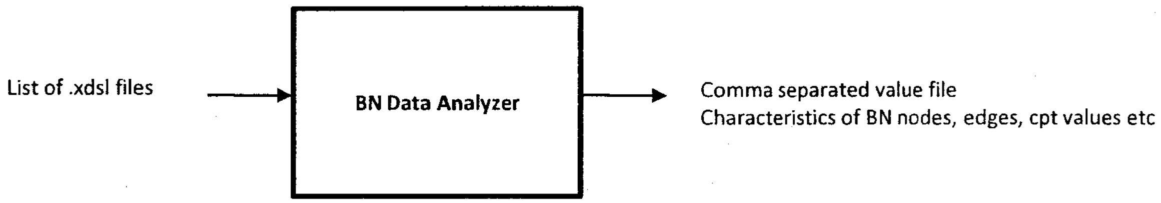

List of .xdsl files
BN Data Analyzer
Comma separated value file
Characteristics of BN nodes, edges, cpt values etc

Figure 37: Simulator - BN Data Analyzer (Sirius)

size and complexity, limited only by the capacity of the operating memory of the computer. GeNIe is a developer environment which produces models that can be embedded or analyzed by any application, such as our simulator.

Graph Visualization Software (Graphviz) Graphviz is an open source graph visualization software [22]. It includes several graph layout programs. It also has web and interactive graphical interfaces, and auxiliary tools, libraries, and language bindings that make it easy for us to visualize attack graphs. The Graphviz layout programs take descriptions of graphs in a simple text language with .dot extension, and produce diagrams in formats such as Postscript or images.

6.2 Simulation Results

The main objective of our simulation studies is to examine the scalability of the proposed models. Although general purpose BN-based models are known to have scalability issues, our observation is that our proposed models are a special class of BNs in the following sense. That is, only a small portion of the conditional probabilities in the BNs are non-trivial and employed to model the logic relationships between exploits and conditions, whereas all other conditional probabilities have the value 0 and thus can be disregarded in storing and analyzing the models. Therefore, through optimizing the storage and analysis techniques in our tools, we expect our models to possess scalability properties that are feasible for implementation when handling attack graphs with node distributions approximating those that can be realistically encountered in real world cases. In the following, we shall present results to confirm this conjecture. In all of our experiments, we use *AG size* for the total number of nodes (exploit or condition) in a given AG, and we use *BN size* for the total number of non-zero conditional probability values (since we do not store the zero values).

BN Size as a Function of AG Size - Normal Distribution For this experiment, the AG generator is set to use a normal distribution for the dependency relationships between exploits and conditions in the AG. Specifically, for this experiment, we have: about 70% of the nodes have 4 incoming edges and 4 outgoing edges, 10% have 1 incoming edge and 4 outgoing edges, 10% have 2 incoming edges and 3 outgoing edges and the remaining 10% have 3 incoming edges and 2 outgoing edges. Figure 38 shows the BN size as a function of the AG size and Figure 39 shows the size of the physical file used to store BN models (.xdsl file) as a function of the AG size.

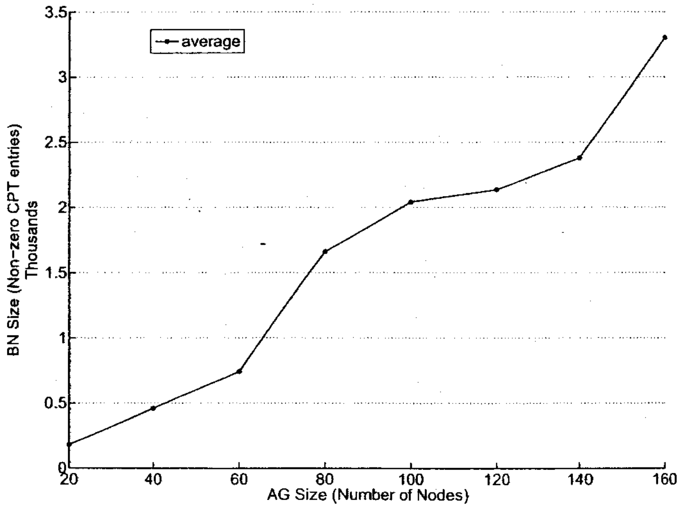


Figure 38: BN Size as a Function of AG Size: Normal Distribution

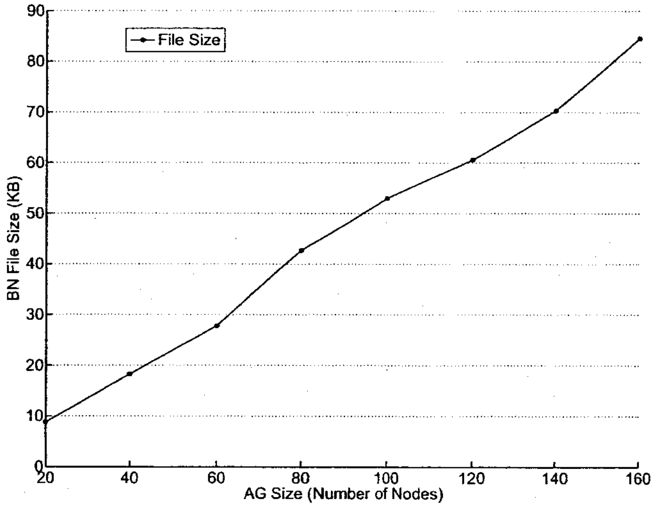


Figure 39: BN File Size as a Function of AG Size: Normal Distribution

From these results, we can conclude that unlike in a general purpose BN model, the number of non-trivial conditional probability values in our BN models does not increase exponentially in the size of the DAG representing the BN (that is, attack graph in our case). The file size actually increases in an approximate linear trend and can be easily managed with modern computers. Therefore, our models can potentially be applied to large networks (however, the generation of attack graphs for large networks has its own scalability issue, which is outside the scope of this thesis).

To show the scalability of our models for attack graphs with different characteristics, Figure 40 and 41 show similar results with dependency relationships that are uniformly distributed. Similar conclusions can be drawn for such attack graphs, that is, our BN-based model is still relatively scalable.

Finally, we study the scalability of our models for attack graphs with different structures. For this experiment, we set the AG generator to fix the total number of nodes in the AG to be 60 but vary the ratio of initial conditions (that is, conditions that are not post-conditions of any exploit) among all nodes. Figure 42 shows the BN size as a function of

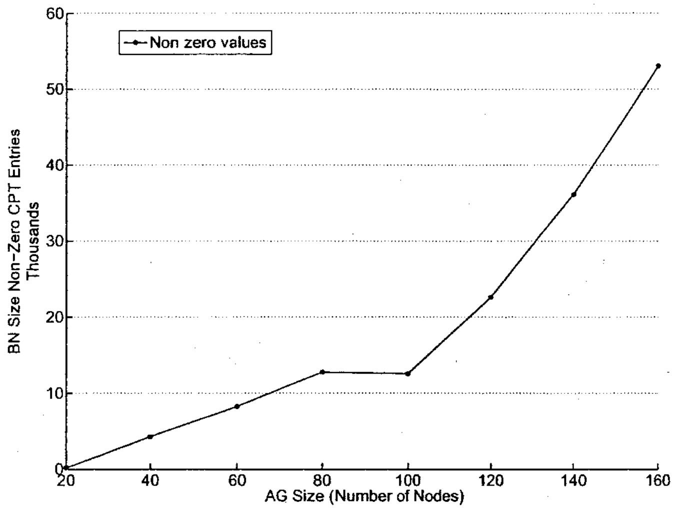


Figure 40: BN Model Size as a Function of AG Size: Uniform Distribution

the ratio of initial conditions. Figure 43 shows the file size (.xdsl file storing the BNs)) as a function of the ratio of initial conditions. These results show that although the structure of attack graphs does have an impact on the size of the BN models, the change is not significant, and our models are still scalable and feasible for implementation.

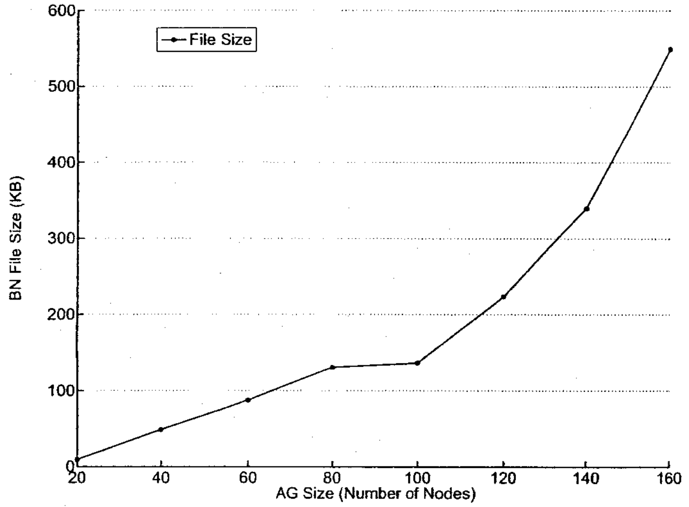


Figure 41: BN Model File Size as a Function of AG Size: Uniform Distribution

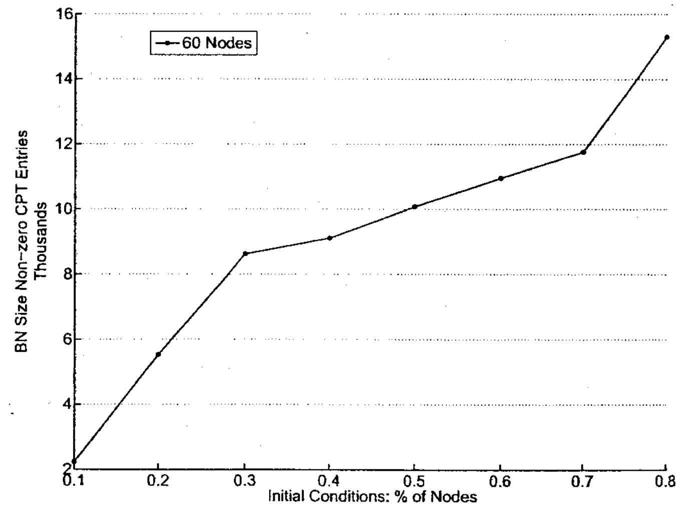


Figure 42: BN Model Size as a Function of AG Size: Initial Conditions

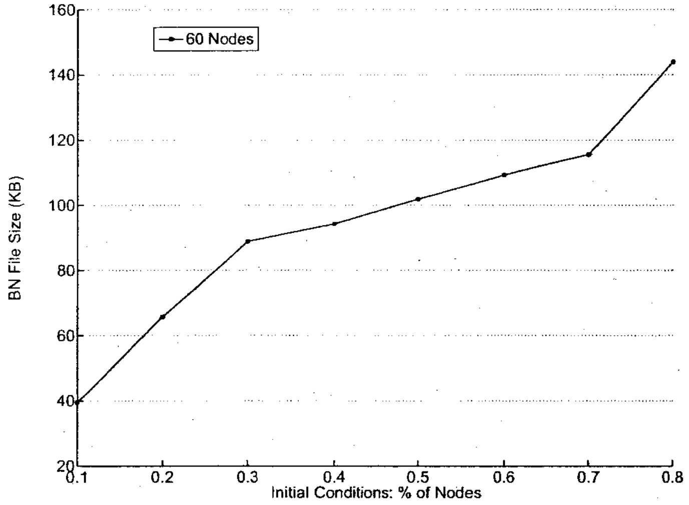


Figure 43: BN Model File Size as a Function of AG Size: Random Distribution

Chapter 7

Conclusion and Future Work

This thesis has pointed out two limitations in existing network security metrics and scoring systems, that is, the lack of consideration of the inter-dependency between exploits and the lack of support for temporal factors. This thesis then proposed a novel BN-based model for addressing such limitations. Specifically, it was shown that BN and DBN can be derived from given AGs and CVSS metric values. The BN model could effectively handle cases where existing approaches to network security metrics fail. The DBN model could be used for useful analysis of the constantly changing security aspects of a network. The fact that our metrics were based on the standard CVSS scores leads to actionable knowledge. The research also provides hints for enhancing the CVSS standard to include mechanisms for combining individual scores and for adjusting temporal metrics based on observed network incidents.

Future research will continue to refine our approach using DBNs to encompass more properties of the temporal metrics established in the CVSS in order to develop a more accurate model. Future research will also examine how the model can be refined to take into consideration the environmental factors of CVSS. We will study the application of the proposed model for hardening a vulnerable network with the least cost. Finally, we will strive to implement and evaluate the proposed models in real world applications.

Publications

Publications related to this thesis are the following:

- M. Frigault, L.Wang, A.Singhal and S. Jajodia. 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.
- M. Frigault, L. Wang, A, Singhal and S. Jajodia. Measuring Network Security Using Dynamic Bayesian Networks. In *Proceedings of the 4th Workshop on Quality of Protection (QoP'08)*, October 2008.

Bibliography

[1] 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)*, 2002.

[2] X. An, D. Jutla, and N. Cercone. Privacy intrusion detection using dynamic bayesian networks. In *Proceedings of the 8th International Conference for Electronic Commerce (ICEC'06)*, pages 208–215, 2006.

[3] Y. Asnar, P. Giorgini, F. Massacci, and N. Zannone. From trust to dependability through risk analysis. In *Proceedings of ARES'07*, 2007.

[4] Applied Computer Security Associates. Workshop on. In *Information Security System Scoring and Ranking*, 2001.

[5] D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In *Proceedings of the 1st Workshop on Quality of Protection*, 2005.

[6] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS'94)*, pages 3–18, 1994.

[7] Common vulnerability scoring system - SIG. available at: http://www.first.org/cvss/, Accessed May 2008.

[8] M. Dacier. Towards quantitative evaluation of computer security. Ph.D. Thesis, Institut National Polytechnique de Toulouse, 1994.

[9] M. Dacier, Y. Deswarte, and M. Kaaniche. Quantitative assessment of operational security: Models and tools. Technical Report 96493, 1996.

[10] University of Pittsburg Decision Systems Laboratory. Genie version 2.0.3259.0 built 03/12/2008. In *Available at http://genie.sis.pitt.edu*, Copyright 1998-2008.

[11] R. Deraison. Nessus scanner, 1999. Available at http://www.nessus.org.

[12] D. Farmer and E.H. Spafford. The COPS security checker system. In *USENIX Summer*, pages 165–170, 1990.

[13] 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.

[14] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of QoP (QoP '08)*, 2008.

[15] Klaus Havelund and Grigore Rosu. Efficient monitoring of safety properties. *Int. J. Softw. Tools Technol. Transf.*, 6(2):158–173, 2004.

[16] J. Homer, A. Varikuti, X. Ou, and M.A. Mcqueen. Improving attack graph visualization through data reduction and attack grouping. In *Proceedings of the 5th international workshop on Visualization for Computer Security (VizSec '08)*, pages 68–79, 2008.

[17] K.S. Hoo. Metrics of network security. White Paper, 2004.

[18] 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 Challenges*. Kluwer Academic Publisher, 2003.

[19] A. Jaquith. *Security Metrics Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley, 2007.

[20] S. Jha, O. Sheyner, and J.M. Wing. Two formal analysis of attack graph. In *Proceedings of the 15th Computer Security Foundation Workshop (CSFW'02)*, 2002.

[21] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attack behaviour. In *IEEE Transactions on Software Engineering, Vol.23 No. 4*, April 1997.

[22] AT&T Research Labs. Graphviz - open source graph layout and drawing software. Available at http://www.research.att.com/sw/tools/graphviz.

[23] Y. Liu and H. Man. Network vulnerability assessment using bayesian networks. In *Proceedings of SPIE - Data Mining, Intrusion Detection, Information Assurance and Data Networks Security (SPIE'05)*, pages 61–71, 2005.

[24] K. Manadhata, J.M. Wing, M.A. Flynn, and M.A. McQueen. Measuring the attack surfaces of two ftp daemons. In *Quality of Protection Workshop*, 2006.

[25] John McHugh. Quality of protection: Measuring the unmeasurable? In *Proceedings of QoP (QoP'06)*, 2006.

[26] V. Mihajlovic and M Petkovic. Dynamic bayesian networks: A state of the art. available at: http://doc.utwente.nl/36632/1/0000006a.pdf.

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

[28] National vulnerability database. available at: http://www.nvd.org, May 9, 2008.

[29] R. Ortalo, Y. Deswarte, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Trans. Software Eng.*, 25(5):633–650, 1999.

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

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

[32] 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*, pages 31–38, New York, NY, USA, 2006. ACM Press.

[33] C. Phillips and L. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the New Security Paradigms Workshop (NSPW'98)*, 1998.

[34] M.K. Reiter and S.G. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security*, 2(2):138–158, 5 1999.

[35] R. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *Proceedings of the 2000 IEEE Symposium on Research on Security and Privacy (S&P'00)*, pages 156–165, 2000.

[36] R. Ritchey, B. O'Berry, and S. Noel. Representing TCP/IP connectivity for topological analysis of network security. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02)*, page 25, 2002.

[37] S. Jajodia. S. Noel. Managing attack graph complexity through visual hierarchical aggregation. In *CCS Workshop on Visualization and Data Mining for Computer SecurityŠ04*, 2004.

[38] S. Jajodia. S. Noel. Understanding complex network attack graphs through clustered adjacency matrices. In *Proceedings of the 21st Annual Computer Security Applications Conference*, 2005.

[39] B. Schneier. Attack trees. *Dr. Dobbs Journal*, 24(12):21–29, 1999.

[40] 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 (S&P '02)*, 2002.

[41] M. Swanson, N. Bartol, J. Sabato, J. Hash, and L. Graffo. Security metrics guide for information technology systems. NIST Special Publication 800-55, 2003.

[42] L. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer attack graph generation tool. In *Proceedings of the DARPA Information Survivability Conference & Exposition II (DISCEX'01)*, 2001.

[43] S. Templeton and K. Levitt. A requires/provides model for computer attacks. In *Proceedings of the 2000 New Security Paradigms Workshop (NSPW'00)*, pages 31–38, 2000.

[44] 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)*, pages 283–296. Springer-Verlag Lecture Notes in Computer Science (LNCS), 2008.

[45] L. Wang, A. Liu, and S. Jajodia. An efficient and unified approach to correlating, hypothesizing, and predicting intrusion alerts. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS 2005)*, pages 247–266, 2005.

[46] L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29(15):2917–2933, 2006.

[47] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 11 2006.

[48] L. Wang, A. Singhal, and S. Jajodia. Measuring network security using attack graphs. In *Proceedings of the 3rd ACM workshop on Quality of protection (QoP'07)*, New York, NY, USA, 2007. ACM Press.

[49] 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.

[50] L. Wang, C. Yao, A. Singhal, and S. Jajodia. Interactive analysis of attack graphs using relational queries. In *Proceedings of 20th IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC'06)*, pages 119–132, 2006.

[51] D. Zerkle and K. Levitt. Netkuang - a multi-host configuration vulnerability checker. In *Proceedings of the 6th USENIX Unix Security Symposium (USENIX'96)*, 1996.