

# GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool

by

Leevar Williams

Submitted to the Department of Electrical Engineering and Computer  
Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2008

© Massachusetts Institute of Technology 2008. All rights reserved.

Author .....  
Department of Electrical Engineering and Computer Science  
May 27, 2008

Certified by .....  
Richard P. Lippmann  
Senior Technical Staff, MIT Lincoln Laboratory  
Thesis Supervisor

Accepted by .....  
Arthur C. Smith  
Chairman, Department Committee on Graduate Students



# GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool

by

Leevar Williams

Submitted to the Department of Electrical Engineering and Computer Science  
on May 27, 2008, in partial fulfillment of the  
requirements for the degree of  
Master of Engineering in Electrical Engineering and Computer Science

## **Abstract**

Attack graphs are valuable tools in the assessment of network security, revealing potential attack paths an adversary could use to gain control of network assets. Creating an effective visualization for attack graphs is essential to their utility, but many previous efforts produce complex displays that are difficult to relate to the underlying networks. This thesis presents GARNET (Graphical Attack graph and Reachability Network Evaluation Tool), an interactive visualization tool intended to facilitate the task of attack graph analysis. The tool provides a simplified view of critical steps that can be taken by an attacker and of host-to-host network reachability that enables these exploits. It allows users to perform “what-if” experiments including adding new zero-day attacks, following recommendations to patch software vulnerabilities, and changing the attacker starting location to analyze external and internal attackers. Users are able to view a set of attack graph metrics that summarize different aspects of overall network security for a specific set of attacker models. An initial user evaluation of GARNET identified problematic areas of the interface that assisted in the development of a more functional design.

Thesis Supervisor: Richard P. Lippmann  
Title: Senior Technical Staff, MIT Lincoln Laboratory



## Acknowledgments

First of all, I would like to thank my advisor Rich Lippmann for guiding me through the completion of my thesis. I am appreciative of the time and effort he has devoted to advising me over the past year, and his consistent encouragement and positive reinforcement have made it a very rewarding experience.

I would also like to thank Kyle Ingols for the invaluable assistance he has given me. I appreciate the great deal of time he has spent coding and debugging to support my work, as well as his constant willingness to offer his help and advice, technical or otherwise. I am grateful to the other members of Group 62 for their useful contributions. In particular, I would like to thank Seth Webster, Rob Cunningham, Tamara Yu, and Chris Connelly for their valuable feedback and suggestions.

Finally, I am thankful for my parents and their unwavering support. They have been a source of encouragement and motivation throughout my academic career and have always believed in my abilities.



# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
1.1	Motivation . . . . .	15
1.2	Goals . . . . .	16
1.3	Overview . . . . .	17
<b>2</b>	<b>Related Work</b>	<b>19</b>
2.1	Attack Graph Visualization . . . . .	19
2.2	Other Visualization Techniques . . . . .	21
2.2.1	Treemaps . . . . .	21
2.2.2	Network Visualization by Semantic Substrates . . . . .	23
<b>3</b>	<b>Background</b>	<b>25</b>
3.1	NetSPA . . . . .	25
3.1.1	Network Model . . . . .	26
3.1.2	Multiple-Prerequisite Attack Graph . . . . .	27
3.1.3	Recommendations . . . . .	28
3.1.4	Security Metrics . . . . .	29
3.2	Attacker Models . . . . .	30
<b>4</b>	<b>User Interface Design</b>	<b>33</b>
4.1	Design Goals . . . . .	33
4.2	Interface Overview . . . . .	34
4.2.1	Visual Representation . . . . .	34

4.2.2	Automatic Layouts . . . . .	36
4.3	Interaction Modes . . . . .	38
4.3.1	Network Map . . . . .	39
4.3.2	Attack Graph . . . . .	41
4.3.3	Summary Plots . . . . .	48
4.4	Timeline . . . . .	50
<b>5</b>	<b>Implementation</b>	<b>55</b>
5.1	Interface to NetSPA . . . . .	55
5.2	GraphModel . . . . .	59
5.3	GraphView . . . . .	60
5.4	MainGui Controller . . . . .	62
<b>6</b>	<b>Evaluation</b>	<b>65</b>
6.1	Testing Methodology . . . . .	65
6.2	Results . . . . .	66
<b>7</b>	<b>Conclusion</b>	<b>71</b>
7.1	Contributions . . . . .	71
7.2	Future Work . . . . .	72
<b>A</b>	<b>User Evaluation Briefing</b>	<b>75</b>
<b>B</b>	<b>User Evaluations</b>	<b>79</b>
<b>C</b>	<b>User Evaluation Summary</b>	<b>97</b>



# List of Figures

1-1	Example attack graph. . . . .	14
2-1	Screenshot of RedSeal Security Risk Manager. . . . .	20
2-2	Example of TVA attack graph display. . . . .	20
2-3	Example tree diagram and corresponding treemap. . . . .	22
2-4	Examples of displays using semantic substrates. . . . .	23
3-1	System architecture of NetSPA tool. . . . .	26
3-2	Example attack graph for simple network. . . . .	28
4-1	Screenshot of GARNET. . . . .	34
4-2	Example of GARNET's attack graph visualization. . . . .	35
4-3	Input dialog for automatic layouts. . . . .	37
4-4	Sample network diagram. . . . .	39
4-5	Subnet Reachability pane and host group context menu. . . . .	40
4-6	Network Information pane. . . . .	40
4-7	GARNET in Attack Graph mode. . . . .	42
4-8	Attacker Depth pane. . . . .	43
4-9	Visualization of one attacker hop. . . . .	43
4-10	Visualization of two attacker hops. . . . .	43
4-11	Recommendations pane and associated visualizations. . . . .	45
4-12	Input dialog for zero-day vulnerabilities. . . . .	46
4-13	Visualization of sample network for 2 attacker models. . . . .	47
4-14	Host Asset Values pane. . . . .	48

4-15	GARNET in Summary Plots mode. . . . .	49
4-16	Metric plots for sample network. . . . .	51
4-17	Metric plots for sample network, with recommendation applied. . . . .	51
4-18	Metric plots for sample network, with vulnerability added. . . . .	51
4-19	Timeline component. . . . .	52
5-1	Module dependency diagram of GARNET. . . . .	56
5-2	Block diagram of interface between NetSPA and GARNET. . . . .	56
5-3	Block diagram of NetSPA's network model objects. . . . .	58
6-1	Original and updated versions of control panel. . . . .	67
6-2	Previous design of timeline component. . . . .	68
6-3	Improved design of timeline component. . . . .	68

# List of Tables

4.1	Description of subnet layouts. . . . .	37
4.2	Mappings of icon color to type of action indicated. . . . .	52



# Chapter 1

## Introduction

As the Internet has grown, organizations have become increasingly reliant on the correct and secure functioning of networked computer systems. System administrators must assess a network's vulnerability to attack, then appropriately modify the network so that it becomes secure. Performing this analysis is a rather difficult task because firewall rules are complex and vulnerabilities on hosts exposed through firewalls are constantly being discovered and patched.

Attack graphs represent one method of analyzing information about a network and its vulnerabilities. They have been proposed by many researchers as a way to identify critical network weaknesses, construct adversary models, evaluate network security, and suggest changes to improve security. Researchers and commercial companies have developed many differing approaches to generating attack graphs — an annotated review of many of these approaches is presented by Lippmann and Ingols [8]. While there are various representations, the overall concept of attack graphs remains the same: they show the ways an attacker can compromise hosts within a network. Conceptually, an attack graph can be thought of as a directed node-link graph. Each node represents a state of the network that an attacker has managed to obtain. Links correspond to actions that allow the attacker to achieve a given state. Figure 1-1 illustrates this particular representation.

Attack graphs are constructed by starting an adversary at a given network location and examining how the attacker can progressively compromise vulnerable hosts, using

information about software vulnerabilities and network reachability. The example in Figure 1-1 shows that an attacker can first use a remotely exploitable buffer overflow vulnerability to achieve administrator-level privileges on a web server. A database server can then be compromised at user level with a remote exploit from the web server, and this privilege can be raised to administrator level by two additional local exploits. Other sequences of exploits are also possible, as indicated by the unlabeled parts of the graph.

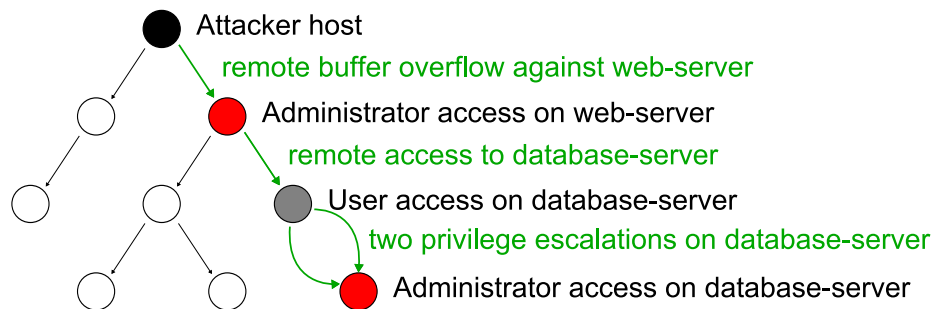


Figure 1-1: Example attack graph.

Previous work presents a system called NetSPA, or Network Security and Planning Architecture, which efficiently generates attack graphs for large complex networks [3, 10]. It imports data from common sources, including vulnerability scanners, firewall configuration files, and vulnerability databases. This information is used to generate attack graphs, make recommendations for improving network security, and compute network security metrics.

NetSPA represents an efficient approach to attack graph generation since it is able to automatically compile all of the necessary information and handle very large networks in reasonable amounts of time. However, it fails to provide an effective display for this information, and other related applications are similarly limited in their ability to visualize attack graphs in a useful and intuitive fashion. GARNET (Graphical Attack graph and Reachability Network Evaluation Tool) is designed as a solution to this problem. It provides a rich interface for visualizing and interacting with attack graphs, and it incorporates various security metrics that can be compared for different networks. This chapter provides the motivation for and an introduction

to GARNET.

## 1.1 Motivation

For a given network, an attack graph shows all hosts that can be compromised by an attacker and the sequences of exploited vulnerabilities and actions that permit these compromises. Although it can be very useful for network security analysis, this method has several practical weaknesses.

Since attack graphs encompass a significant amount of data about a network, they quickly become overloaded with information. The number of paths in a full attack graph [10] can grow combinatorially as the amount of hosts increases, and for practical networks with hundreds or even dozens of hosts, the corresponding graphs can be large, complex, and difficult to visually interpret. Even if simplifications are applied (e.g. grouping similar hosts and attacker actions into single nodes and links), the resulting graphs would still be unmanageably large for modest numbers of hosts. As a consequence, manual analysis of the graphs is nearly impossible, making it difficult to extract useful information.

Another limitation of attack graphs is their inadequate representation of host-to-host reachability. Reachability is a key factor in explaining why certain attacker paths are possible and in determining which filtering devices allow these attack steps and could be modified to prevent them. The connectivity amongst hosts in a network is not explicitly represented, but it certainly influences the structure of the attack graph. The integration of this information with attack graphs would facilitate the understanding of a network's overall security posture.

Finally, attack graphs do not inherently reflect the topology of the underlying network. In a physical network, hosts are partitioned into fully connected domains such as subnets and virtual local area networks by firewalls and other filtering devices. Furthermore, these host domains have some natural arrangement, either physical or conceptual, that would be depicted in a diagram of the network. The structure of an attack graph does not necessarily conform to these intuitive groupings, thus making

it difficult for a system administrator to relate the graph to the actual network.

These shortcomings motivate the need to develop an attack graph visualization that is scalable, informative, and interactive. Such a display would improve a network administrator's ability to leverage the data contained within an attack graph and take actions to effectively secure a network.

## 1.2 Goals

GARNET's visualization of the network and attack graph addresses all of the limitations described above and is designed to achieve the following goals:

- **Partition the network elements into groups.** Hosts and other components on the network are divided into fully connected domains, as defined by the filtering devices such as firewalls and routers, and are displayed within separate groups. This grouping aids in conveying the overall structure of the network.
- **Illustrate general network reachability.** The reachability between hosts is incorporated into the display to help users better understand the availability of paths into the network that can be exploited by an attacker.
- **Highlight critical attack paths.** Placing an emphasis on critical, worst-case paths in the attack graph encourages users to initially focus on the more severe potential threats. This reduces the complexity of the full graph that causes it to be virtually indecipherable.
- **Incorporate automated recommendations and security metrics.** Large, heterogenous networks can produce a complex set of attack paths, regardless of simplifications made to the graph. NetSPA's automatically generated recommendations suggest sets of vulnerabilities that should be patched, and the security metrics it computes provide a concise summary of the security of the network. This information, produced by automatic analysis of the attack graph, is intended to complement and promote comprehension of the visual display.



- **Allow direct interaction with the attack graph.** The display supports direct manipulation of the attack graph, enabling users to create their own intuitive layout. The visibility of certain types of information about the network and attack graph can also be interactively controlled. This way, users can vary the level of detail displayed to avoid being overwhelmed by irrelevant details.
- **Enable the simulation of modifications to the network.** Because of NetSPA's ability to rapidly compute attack graphs, the visualization can be dynamically regenerated to reflect arbitrary changes to the network. System administrators are therefore able to experiment with hypothetical networks and types of attackers and receive immediate feedback about the effects these actions would have on the attack graph. As a consequence, different sets of preventative actions could be evaluated before they are applied, and certain vulnerabilities could be simulated to identify other potential threats.

## 1.3 Overview

The remainder of this thesis describes GARNET's design, implementation, and evaluation in detail. The following chapter presents an overview of related work involving attack graph displays and other types of graph visualization. Chapter 3 describes the NetSPA system and provides background information on the types of attackers that can be modeled in GARNET. In Chapter 4, GARNET's user interface is described, including its main display and modes of interaction. Chapter 5 examines details of the tool's implementation. This is followed by a discussion of the user evaluations conducted on the interface in Chapter 6. Finally, Chapter 7 presents the contributions of this thesis and discusses possible directions for future work.



# Chapter 2

## Related Work

There is currently a substantial body of research in the area of attack graph generation and analysis, and much of it has focused on the problem of data visualization. GARNET builds upon some of this previous work and also draws from general graph and network visualization techniques that have been used in other applications. This chapter gives an overview of relevant research and software tools.

### 2.1 Attack Graph Visualization

A variety of approaches have been developed that focus on visualizing and interacting with attack graphs. Recently, two commercial companies have integrated attack graph displays into their risk management products. The RedSeal Security Risk Manager [22] reads vulnerability information from network vulnerability scanners and topology information from firewall and router configuration files to create a node-link network topology diagram. The tool identifies exploitable vulnerabilities and, on top of the network diagram, displays threat paths that an attacker can use to gain access to resources in the network. It also proposes a prioritized list of vulnerabilities to be fixed. A screenshot of the topology view with overlaid attack paths appears in Figure 2-1.

The second commercial product, Skybox View [25], provides a similar network view. However, it requires active agents to capture network topology and host vul-

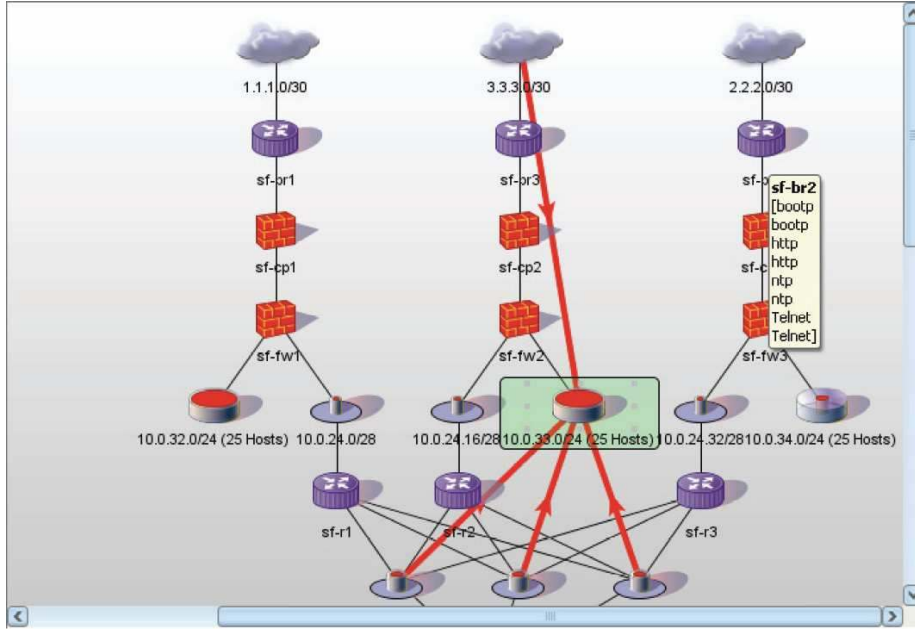


Figure 2-1: Screenshot of RedSeal Security Risk Manager’s network topology diagram with overlaid attack paths.

nerability information. Reachability is computed and attack paths are shown in a separate display as arrows between individual hosts or servers. The application allows “what-if” analyses to be performed through the simulation of attacks and proposed changes to the network. It is limited, though, by the fact that it does not show the entire attack graph but only displays parts of the overall graph that contain specified target hosts.

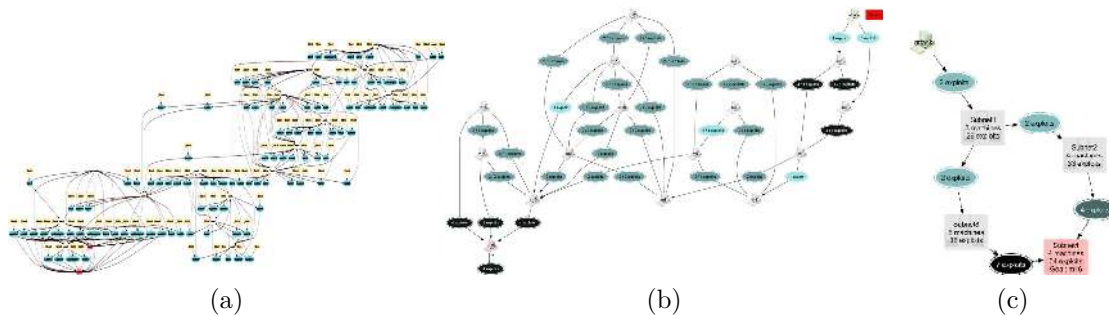


Figure 2-2: Example of TVA attack graph display, where the graphs in (a) through (c) have been progressively clustered.

Jajodia and Noel describe a tool that uses an approach known as Topological Vulnerability Analysis (TVA) [4]. Their TVA system generates a model of the network

and builds an attack graph based on a user-defined attack scenario, which specifies an attacker starting point, the attack goal(s), and hypothetical changes to the network configuration. The display uses graph clustering methods to simplify the original node-link attack graph into a more compact form. There are several levels of clustering, allowing the graph to be viewed at different levels of detail. Figure 2-2a shows a sample TVA display of a full attack graph. In Figure 2-2b, the original graph has been clustered to contain nodes that represent individual machines and links that show exploits between them. Finally, Figure 2-2c shows a further simplified graph whose nodes correspond to entire subnets.

Another technique developed by Noel and Jajodia [18] focuses on visualizing network reachability as well as attack graphs. It assumes that host-to-host reachability is provided and computes attack graphs from reachability and vulnerability data. The visualization consists of adjacency matrices whose rows and columns correspond to host machines. Each cell would then be assigned a grayscale value that represents either reachability or an attack path between two hosts. The rows and columns are ordered to reveal meaningful clusters in the data. This approach represents a departure from the traditional node-link graph display that may be difficult for system administrators to understand.

## 2.2 Other Visualization Techniques

Graph and network visualization has been one of the most studied areas in the field of information visualization. An excellent review of past approaches is presented in a talk by Munzner [14]. For these visualizations, much of the difficulty lies in finding effective layout algorithms for large graphs. The two techniques described in this section seek to manage the complexity of traditional node-link graph displays.

### 2.2.1 Treemaps

Treemaps, developed by Johnson and Shneiderman [6, 23], are used to display hierarchical information structures that can be represented as trees. This visualization

is a space-filling approach, and it subdivides a rectangular region into smaller rectangles that correspond to the leaf nodes of the tree, which may be assigned different weights. The subdivision produces clusters that mirror the tree’s hierarchical structure, while the size of the rectangles conveys the relative weights of the nodes. The general algorithm is as follows: 1) for each of the direct children of the root node, the input rectangle is divided into vertical sections that are proportional to the combined weights of each child’s subtree; 2) then, for each subtree, the corresponding rectangular slice is horizontally partitioned according to the same procedure. This operation is recursively applied to each node in the tree; the nodes are therefore split vertically at even levels and horizontally at odd levels. Figure 2-3 illustrates this concept. Figure 2-3a shows a tree diagram where the nodes are labeled with a letter and an associated weight. In Figure 2-3b, the treemap is depicted.

Treemaps are not particularly well-suited to conveying hierarchical tree structures since it can be difficult to recognize the hierarchy for large trees. They do, however, result in an efficient use of space and are useful in representing certain attributes of the tree elements. The relative weights are explicitly represented, and color and shading can be used to illustrate additional features. Other layout algorithms are also possible, and Bederson et. al [1] discuss several techniques that attempt to minimize the aspect ratios of the rectangles (i.e. to make them as “square” as possible).

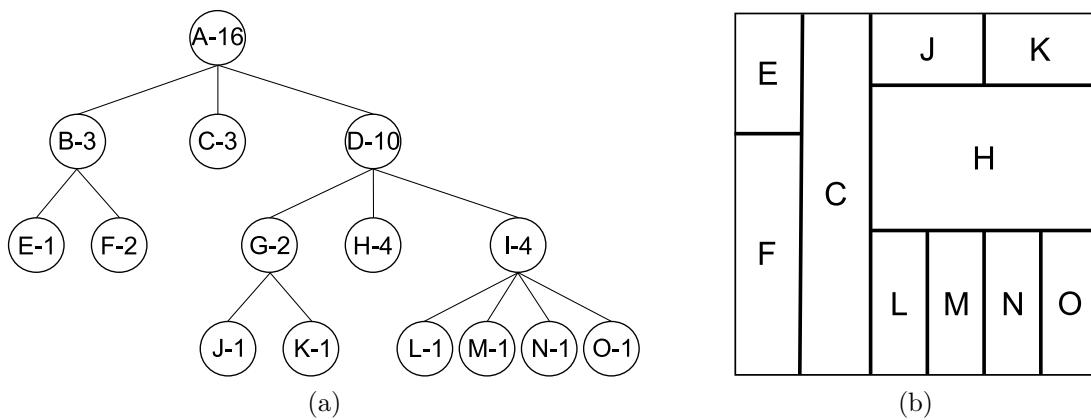


Figure 2-3: Example of tree diagram (a) and its corresponding treemap (b).

## 2.2.2 Network Visualization by Semantic Substrates

In previous research, Shneiderman and Aris [24] describe a visualization approach that is scalable and limits the clutter seen in large, highly-connected graphs. The node layout is based on *semantic substrates*, which are non-overlapping regions defined by the user that correspond to specified attributes of the data. Nodes are grouped according to these substrates; further attributes of the nodes can be encoded by their color, size, and 2D position. The second feature of this visualization is the ability for users to interactively control link visibility. They may choose to only view links within a particular region, across regions, or between nodes within a particular range of attribute values. Similarly, nodes can be displayed or hidden based on their characteristics. In Figure 2-4, there is an example of a display with two substrates and no links drawn (Figure 2-4a), and one containing three substrates with outgoing links shown from nodes in the bottom group (Figure 2-4b). The relative sizes of the regions indicate the number of nodes they contain, and links that cross regions can be quickly distinguished.

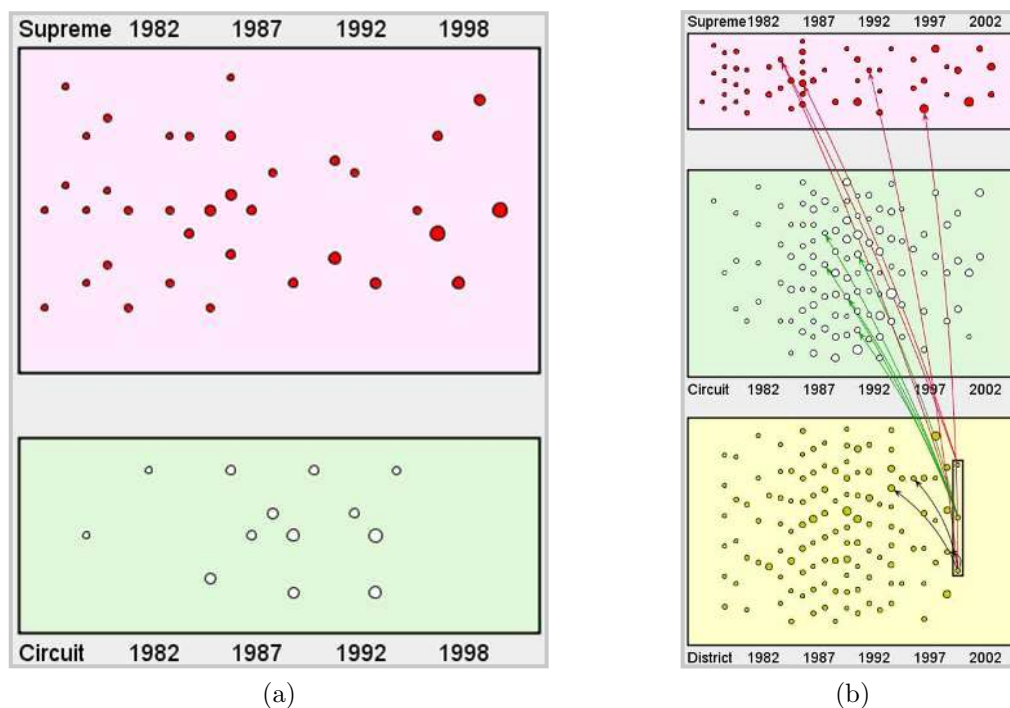


Figure 2-4: Two examples of displays using network visualization by semantic substrates.

The semantic substrates technique can effectively represent large graphs since the nodes are partitioned into high-level groupings and the complexity of the graph can be explicitly controlled by specifying the nodes and edges to display. This interaction enables users to navigate a relatively uncluttered display, with additional information only being shown on demand.



# Chapter 3

## Background

### 3.1 NetSPA

The NetSPA system used by GARNET generates a type of attack graph known as a multiple-prerequisite (MP) graph that scales well to large enterprise networks [3, 9, 10]. NetSPA is comprised of several software components. The importer, written in PERL, is responsible for reading in raw data such as Nessus scans [15], firewall rulesets, the CVE dictionary [2], and NVD database records [19], and converting the data into a custom binary file format for later use. A small C program acts as a vulnerability classifier and is designed to identify a vulnerability's locality (where it can be exploited from) and effect (the privilege level it provides on a target host). It uses a pattern-matching algorithm that has been trained on a sample vulnerability data set. The classifier was built using the freely available LNKNet tool [11]. The computation engine, written in C++, reads the network model from the custom binary file generated by the importer. It is responsible for computing reachability, constructing attack graphs, and analyzing the graphs to generate recommendations and compute security metrics. The block diagram in Figure 3-1 gives an overview of the design of the NetSPA system.

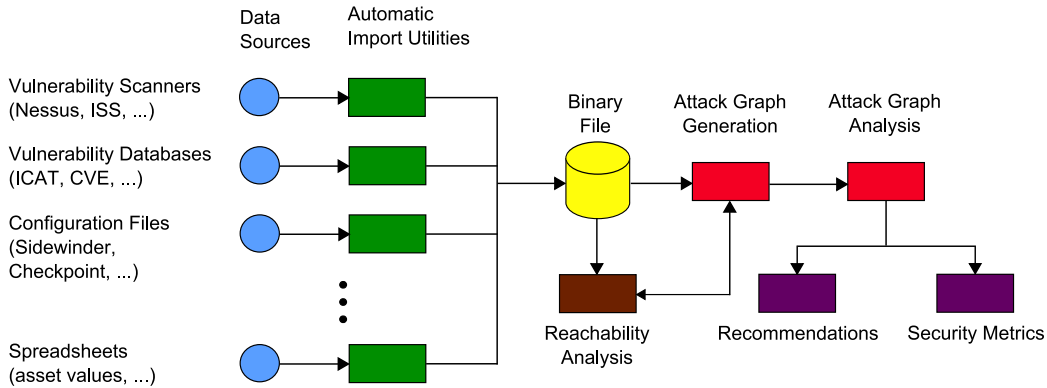


Figure 3-1: System architecture of NetSPA tool.

### 3.1.1 Network Model

NetSPA models both hosts and network infrastructure devices such as firewalls and routers. It assumes that hosts can have one or more open ports that accept connections from other hosts and that ports have zero or more vulnerabilities that may be exploitable by an attacker. Individual vulnerabilities provide one of four access levels on a host: *root* or administrator access, *user* or guest access, *DoS* or denial-of-service, or *other*, indicating a loss of confidentiality and/or integrity. Vulnerabilities can either be exploited remotely from a different host or only locally from the vulnerable host. If *root* or *user* access is achieved, it is assumed that the attacker gains full access to a host and can use it to stage further attacks. Attackers can also obtain credentials when compromising a host. Credentials refer to any information that can be used to gain access to another host or other network resources such as a password or a private key, and they are used to model some kinds of trust relationships. Reachability and credentials serve as prerequisites for exploitation of other vulnerabilities.

NetSPA also incorporates a simple model of host asset values. Each host is assigned an asset value representing the worth to an attacker of the worst-case compromise of that host's confidentiality, integrity, or availability. Asset values currently default to 10 for all hosts and are typically hand-assigned to higher values for critical hosts, such as servers or hosts containing confidential information. They are primarily used for prioritizing recommendations and computing security metrics.

### 3.1.2 Multiple-Prerequisite Attack Graph

The multiple-prerequisite attack graphs generated by NetSPA are fast to compute and can model concepts such as credentials and trust relationships. The graphs have countless edges and use three types of nodes:

- *State* nodes represent an attacker’s level of access on a particular host. Outbound edges from state nodes point to the prerequisites they are able to provide to an attacker.
- *Prerequisite* nodes represent either a reachability group or a credential. Reachability groups represent a collection of reachable ports on the network — these are determined automatically by firewall rule analysis. A credential can be used to model many types of trust relationships such as shared passwords used to administer many hosts or automated remote host control. Outbound edges from prerequisite nodes point to the vulnerability instances that require the prerequisite for successful exploitation.
- *Vulnerability instance* nodes represent a particular vulnerability on a specific port. Outbound edges from vulnerability instance nodes point to the single state that the attacker can reach by exploiting the vulnerability.

These three node types define the sole ordering of paths in the graph: a state provides prerequisites, which allow exploitation of vulnerability instances, which in turn provide more states to the attacker. MP graphs are limited in size because they contain at most one node for each vulnerability instance, host state, reachability group, and credential.

Figure 3-2a shows a simple example network where an attacker begins on host A. Every other host has a single unique, remotely-exploitable vulnerability, and the firewall only permits communication from hosts C and D to host E. The MP attack graph for this network appears in Figure 3-2b. In this illustration, the circles are state nodes and the triangles correspond to vulnerability instance nodes. The rectangles are prerequisite nodes which, in this case, represent reachability. All of the possible

attack paths for the network are implicitly represented by the cyclical edges, making the MP graph much more compact than a full graph that explicitly shows all paths.

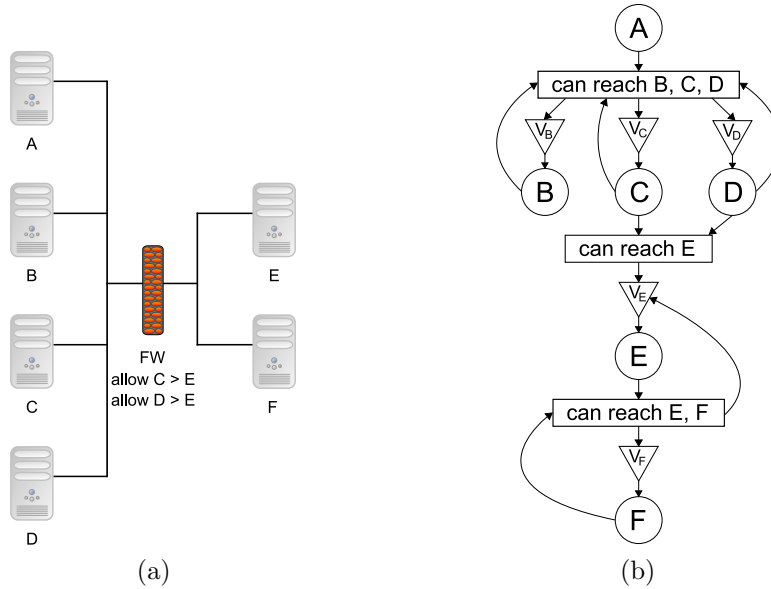


Figure 3-2: A simple example network (a), and its MP attack graph (b).

Although MP graphs scale well compared to other graph representations, they can still be large and contain numerous cycles. A simple algorithm to “collapse” many nodes together was developed in order to simplify the graph’s structure [3]. First, state nodes are combined when their access levels match, and when they both require and provide the same prerequisites. Vulnerability instance nodes are then combined when the prerequisites necessary to exploit them and the states they provide are identical. The resulting collapsed graph shows the relationships between prerequisites and the quantity of compromise they enable.

### 3.1.3 Recommendations

The collapsed MP graph structure is very scalable and greatly reduces the complexity of the attack graph. Nevertheless, the graph’s structure is not well-suited to human interpretation, and manual analysis of even the collapsed graphs is nearly impossible for substantial networks.

NetSPA remedies this problem by automatically analyzing the attack graphs it

generates and creating a list of recommendations. Each of these recommendations identifies a set of vulnerability instances that must be patched to protect a certain set of hosts. They are formed by computing, for each prerequisite in the attack graph, which vulnerability instances must be removed in order to prevent the attacker from obtaining the prerequisite and which states would no longer be reachable without it. NetSPA regenerates the graph for each recommendation to determine the number of hosts that are protected as a result of removing the particular vulnerabilities, and they are prioritized by the total asset value of the protected hosts. The resulting list of suggested patches helps in identifying critical, bottleneck hosts; an example of this would be a directly-accessible host that an attacker must compromise through a firewall in order to gain access to a larger portion of the network.

### 3.1.4 Security Metrics

In addition to producing recommendations, NetSPA’s automated analysis of the attack graphs also outputs a set of security metrics that represent different aspects of attacker effort. This analysis addresses a major shortcoming in the security field, and NetSPA attempts to compute metrics that: are accurate, objective, and well defined; can be measured automatically; are easy to understand and explain; and provide insight into underlying causes of both security and insecurity. The measurements provide a basis on which the overall security of complex networks can be compared — they can indicate how security has changed and if a given network is more or less secure than another. NetSPA generates three security metrics that all depend on the asset values that are assigned to hosts. For each metric, the total amount of assets captured by the attacker is calculated as a function of one of the following measures of attacker effort: attacker hops, attack complexity, and number of unique exploits required.

The first security metric computes the amount of assets captured after each successive attacker hop. A hop indicates the set of hosts that become compromised at the corresponding depth in the attack graph. For example, the first hop represents the hosts that can be directly compromised from the adversary’s starting location.

Each subsequent attacker hop represents those hosts that can be compromised from the set of already compromised hosts. Each hop indicates extra work on the part of the attacker because it can delay capture of the most important assets, require more involved attacks, and provide more opportunity for detection.

The second measure of attacker effort examines the cumulative complexity of attacks required to exploit vulnerabilities on a set of target hosts. The value used for this metric is defined in the “Access Complexity” field of the Common Vulnerability Scoring System [12]. Each exploit that the attacker can use to compromise a host is assigned a numerical value representing either a “low complexity” exploit, such as a buffer overflow that is easily exploitable, or a “high complexity” attack, such as one that requires a rarely seen configuration or involves social engineering methods. These effort values are arbitrarily scaled, and low-cost and high-cost exploits are given values of 20 and 2000, respectively.

The final metric computed by NetSPA measures the number of unique exploits required by an adversary to capture network assets, and it assumes that it requires significantly more effort to obtain or develop a new exploit than to reuse an existing one. The metric involves computing the number of unique exploits that is needed for an attacker to maximally compromise a network. One approach to computing this number would be to find the optimal set of exploits used by an omniscient attacker with full knowledge of the network. Practically, however, attackers have a “limited horizon,” meaning they can only probe hosts that are reachable from currently compromised hosts. At each attack step, an exploit is randomly chosen from the set of exploits that have not yet been used and that allow the compromise of one or more currently reachable hosts. Many sequences of exploits are produced this way, representing the range of capabilities for a limited-horizon attacker.

## **3.2 Attacker Models**

NetSPA employs a simple vulnerability model in its computation of an attack graph, and it assumes that an attacker knows about all existing vulnerabilities and will

exploit them to the fullest extent. Since GARNET allows interaction with the network and attack graph, this basic model can be extended to include other, more capable attackers. GARNET supports experimentation with three primary attacker models, summarized below.

- A *simple attacker* has exploits for all known vulnerabilities. Since exploits are often available on the Internet a few days after a vulnerability is announced [26], this represents an adversary who downloads exploits at low cost but is not able to create new exploits. This is NetSPA's default model.
- A *single-zero-day attacker* has all the exploits of the simple attacker but is also able to create or buy one exploit for a currently non-public vulnerability. This represents a more capable adversary who can craft one zero-day exploit specifically designed to penetrate the network being analyzed.
- A *comprehensive-zero-day attacker* models an adversary that is assumed to have an exploit for every open port in the network. This provides an upper bound on the percentage of network assets that can be captured by extremely capable adversaries. It is assumed that these attackers can determine the vulnerabilities on all hosts reachable from currently compromised hosts and that: they are not concerned about being detected by intrusion detection systems or other network monitors; they are only interested in the cost of developing and launching attacks; and their goal is measured by the total asset value of all hosts compromised.





# Chapter 4

## User Interface Design

### 4.1 Design Goals

As previously mentioned in Chapter 1, GARNET was developed to achieve several goals:

- Partition the network elements into groups
- Illustrate general network reachability
- Highlight critical attack paths
- Incorporate automated recommendations and security metrics
- Allow direct interaction with the attack graph
- Enable the simulation of modifications to the network

The chapter provides an overview of GARNET's user interface. The following sections describe how the attack graph visualization and the interactions it affords support the aforementioned goals.

## 4.2 Interface Overview

The GARNET user interface consists of a main display area, with a control panel on the left side and a timeline control at the bottom. A group of buttons located above the control panel is used to switch between the three modes of interaction. Consistent with other software applications, a menu bar appears at the top of the window and presents the user with view options and a standard set of file functions. Figure 4-1 indicates the major components of the interface.

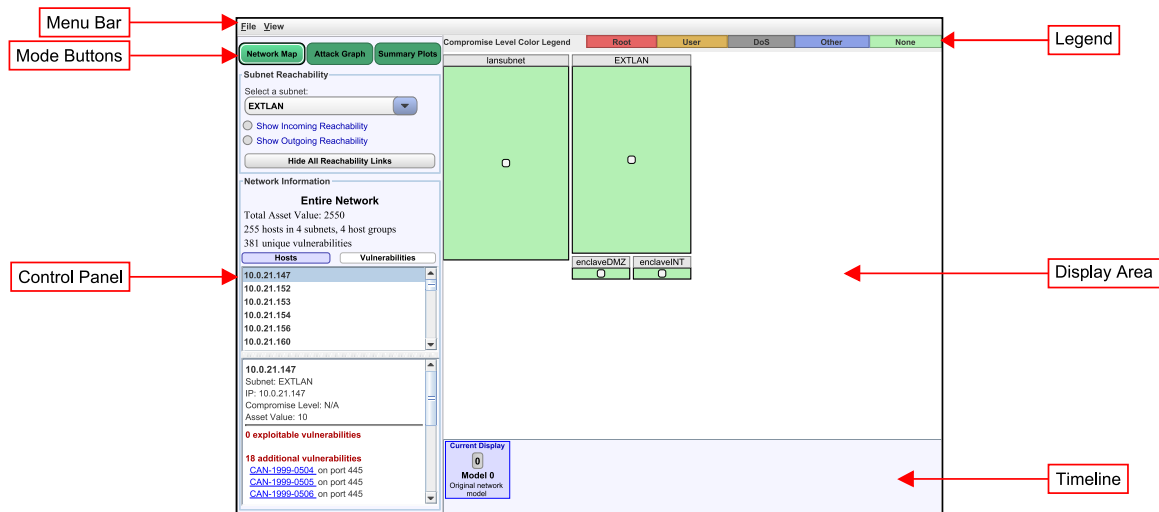


Figure 4-1: Screenshot of GARNET, with its major components labeled.

### 4.2.1 Visual Representation

GARNET's visualization of the MP attack graphs produced by NetSPA is readable and concise, while preserving much of the essential information. Important features of the nodes are conveyed by grouping, size, and color, while other attributes and edge information are initially hidden and can be displayed on demand. This approach is inspired by the semantic substrate displays described in Section 2.2.2.

Nodes from the attack graph are grouped by subnet and a treemap layout is used to illustrate these groupings. Subnets are represented within the display by rectangular blocks labeled with the name of the subnet. Smaller rectangles within each block symbolize host groups; these correspond to the collapsed state nodes from

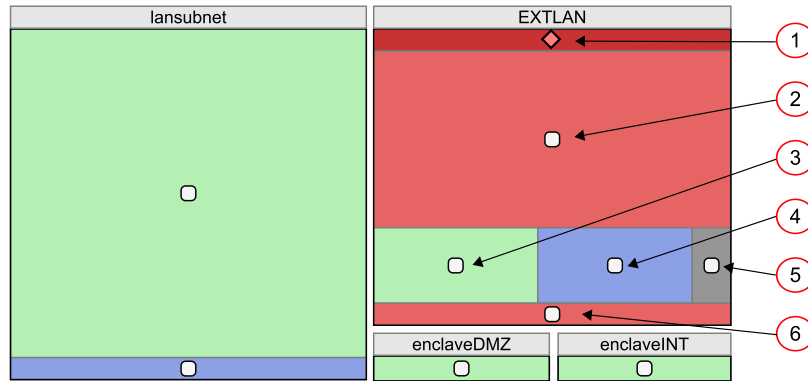


Figure 4-2: Example of GARNET’s attack graph visualization.

the MP attack graph, as detailed in Section 3.1.2. The hosts in a host group are all in the same subnet, compromisable at the same level, and treated identically by all network filtering devices. Each group is colored according to its level of compromise, and the relative size of a host group is proportional to the number of hosts it contains. A white dot also appears in the center of each host group. While the outer rectangle indicates compromise level and relative size, the inner dot functions as a control for selecting the host group and displaying a context menu for node-specific information and functions.

Figure 4-2 provides an example of this visualization for a network with 4 subnets: an external subnet labeled “EXTLAN” containing 119 hosts, an internal subnet labeled “lansubnet” containing 129 hosts, and two smaller subnets containing one host each. The meanings of the six labeled subgroups in the “EXTLAN” subnet are given below.

1. The attacker starting location is indicated by the dark red band with a pink diamond in the center, located at the top of the group.
2. The large red rectangle below shows a group of hosts compromised at the *root* level.
3. The light-green rectangle in the third row represents hosts that cannot be compromised.
4. The blue rectangle to the right represents hosts compromised at the *other* level.

5. The small gray rectangle represents hosts compromised only at the *DoS* level.
6. The red rectangle at the bottom of the group represents hosts also compromised at the *root* level, but with different reachability to other hosts than those of the upper red rectangle (2).

The rectangles in the other subnet groups of Figure 4-2 have similar meanings. A legend that shows each color and its corresponding compromise level is displayed at the top of the display area, as labeled in Figure 4-1. It is minimally obtrusive and is constantly available for quick reference.

Rectangles within each subnet group are laid out according to the strip treemap algorithm [1]. Treemaps were employed for this visualization because they solve the bin-packing problem of completely filling a rectangular region with boxes of different areas. Furthermore, this particular “strip” layout was chosen because, unlike many of the other layout algorithms described by Bederson et al. [1], it produces dimensions with reasonable aspect ratios and accurately conveys ordered data.

## 4.2.2 Automatic Layouts

Within the display area, subnet groups can be repositioned and resized by direct manipulations, i.e. clicking and dragging with the mouse. A user can thus place subnets into an arrangement that represents a physical or similarly intuitive view of the network. However, this manual repositioning can be a tedious task for dozens of subnets, and a user would quickly become frustrated if forced to create a useful layout for each different network they loaded into the tool.

As a solution, the interface provides a variety of automatic layouts. This feature can be accessed through the “View” menu, and a dialog appears with the available options, as illustrated in Figure 4-3. In addition to the three layout types, the user can also select a sizing option that adjusts the sizes of the subnet rectangles to be either uniform or proportional to the number of hosts they represent. Table 4.1 describes the resulting subnet arrangement for each combination of layout and sizing.

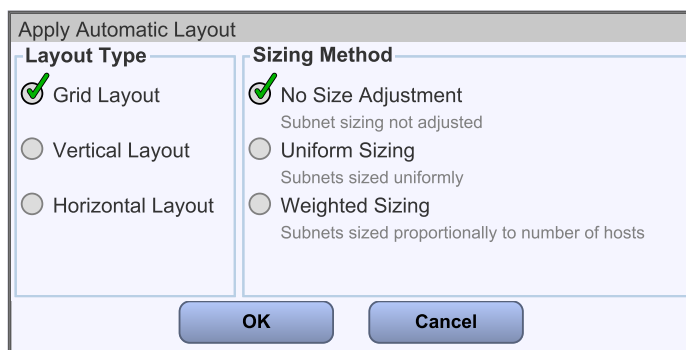


Figure 4-3: Dialog allowing user to select an automatic layout.

	No Sizing	Uniform Sizing	Weighted Sizing
Grid Layout	Subnets are arranged according to the nearest-fit decreasing height algorithm.	Subnets are arranged into a uniform grid, whose dimensions match those of the display area as closely as possible.	Subnets are arranged according to the squarified treemap algorithm, which automatically computes the dimensions and positions.
Vertical Layout	Subnets are tiled vertically along the left side of the display area.	Subnets are sized uniformly and tiled vertically along the left side of the display area.	Subnets are tiled vertically along the left side of the display area. The dimensions are computed by the squarified treemap algorithm.
Horizontal Layout	Subnets are tiled horizontally along the top of the display area.	Subnets are sized uniformly and tiled horizontally along the top of the display area.	Subnets are tiled horizontally along the top of the display area. The dimensions are computed by the squarified treemap algorithm.

Table 4.1: Descriptions of the arrangements that are produced for each combination of layout type and sizing method.

The horizontal and vertical layouts are trivial to produce, but the grid layouts require slightly more involved algorithms since it is not a straightforward task to arrange arbitrarily-sized rectangles into a bounded area. The nearest-fit decreasing height algorithm [13] used for the grid layout without size adjustment is a simple method that tiles the rectangles (ordered by decreasing height) across the display in rows, creating a new row whenever the current one reaches the edge of the window. The squarified treemap layout [1] sizes the rectangles to be as square as possible, then arranges them to fill the entire area of the display window.

The layout settings that are chosen from the dialog are saved in a submenu that also appears in the “View” menu, so that users can quickly access frequently used settings. Further, the interface is capable of creating layout files by translating the current subnet layout into an XML format and writing it to disk. A user therefore has the ability to customize one of the automatic layouts and save this configuration, which can then be loaded from file at a later time and applied to the same network.

When a network is initially loaded, its default subnet layout is a grid arrangement with weighted sizing. This initial configuration is useful because it clusters the subnets and quickly conveys their relative sizes and the overall scale of the network. The grouping of the subnets in Figure 4-2 illustrates this default layout.

### 4.3 Interaction Modes

GARNET’s user interface supports three separate modes of interaction: Network Map, Attack Graph, and Summary Plots. The user toggles between these different modes by clicking one of three mode buttons located in the upper left of the control panel, as indicated in Figure 4-1. Each mode differs in the information that is available and the ways in which the display can be manipulated. The following sections will describe each in detail, and all subsequent examples of the tool’s usage will be based on the sample network in Figure 4-4.

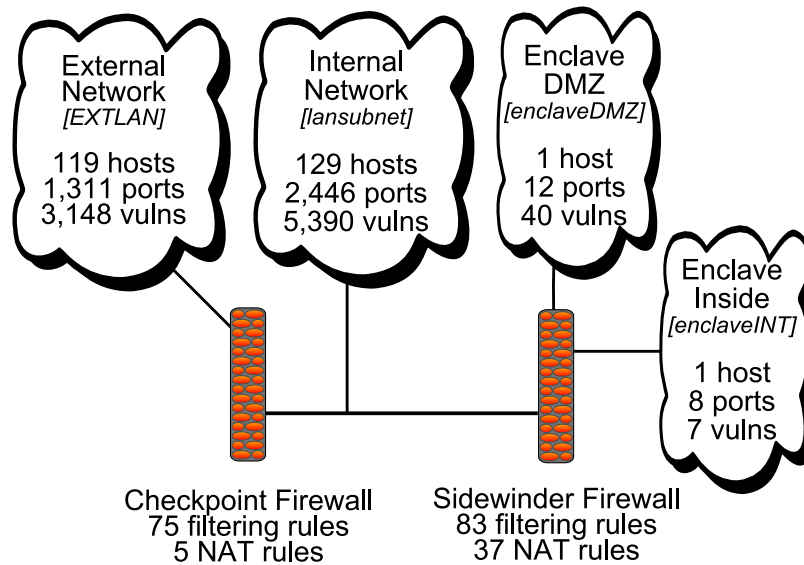


Figure 4-4: Diagram of sample network to be used in subsequent examples.

### 4.3.1 Network Map

The Network Map mode provides an overview of the network's topology, hosts, and vulnerabilities. The screenshot in Figure 4-1 shows GARNET in Network Map mode. In this view, the display area contains the visualization discussed in the previous section, and the control panel is divided into two panes.

#### Subnet Reachability

The first "Subnet Reachability" pane, shown in Figure 4-5a, contains controls for displaying reachability between subnets. The user can select a subnet from a drop-down list or directly by clicking on its rectangle, and options are presented for showing incoming and outgoing connections for all of the host groups within that subnet. This functionality can also be accessed for individual host groups through their context menus; hovering the mouse over the white dot for a host group will cause the menu to appear (Figure 4-5b). The reachability amongst host groups is illustrated by directed edges drawn between pairs of groups. These edges indicate that the hosts in the source group can connect to ports on hosts in the target group.

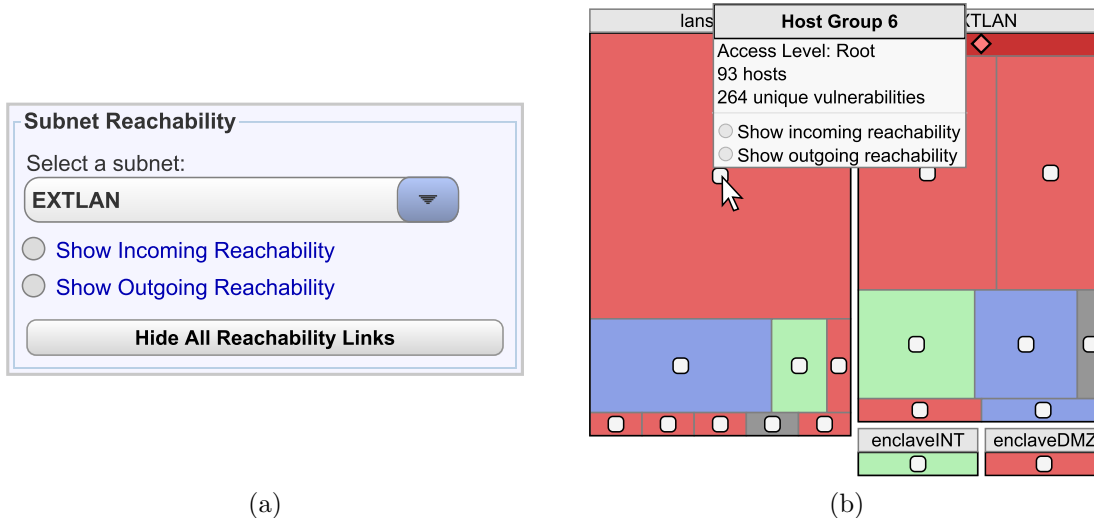


Figure 4-5: Views of the Subnet Reachability control pane (a) and the context menu for an individual host group (b).

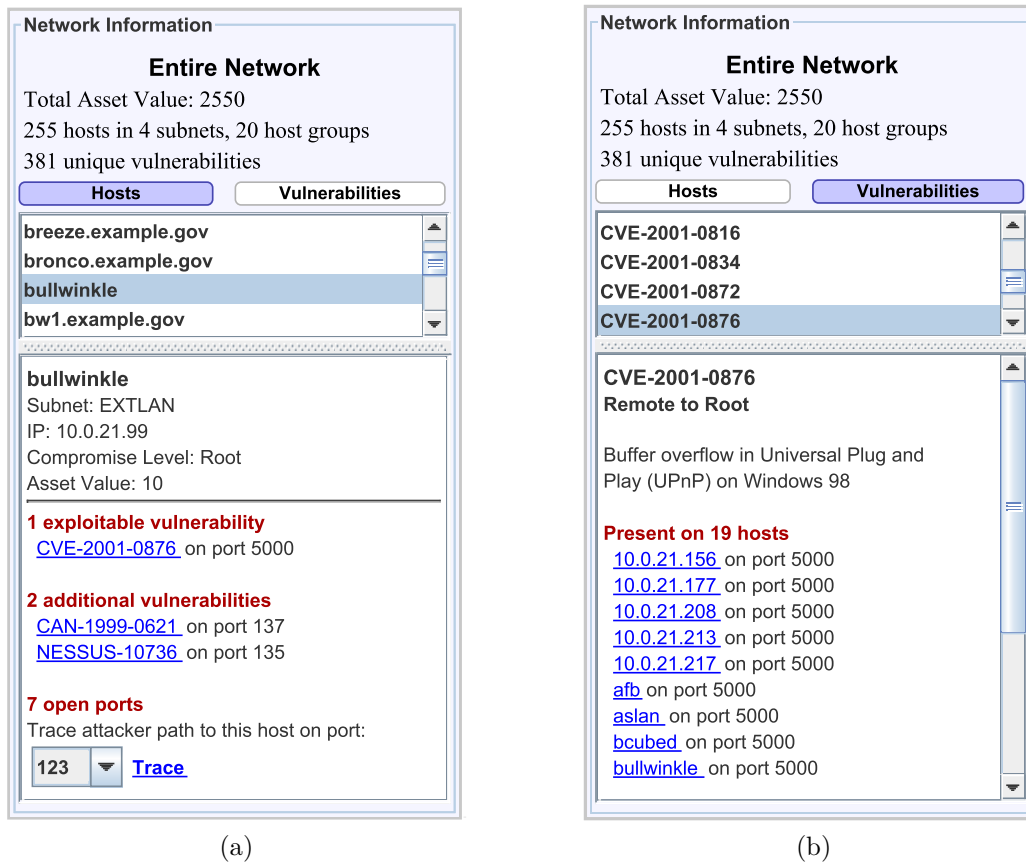


Figure 4-6: Views of the Network Information pane showing the list of hosts (a) and the list of vulnerabilities (b) for the entire network.



## Network Information

The second control pane, labeled “Network Information”, contains an information box that lists hosts and vulnerabilities for the network. A particular host group or subnet can be selected directly from the display by mouse click, and the upper section of the pane indicates the selection, along with a summary of its properties. If nothing is selected, the properties of the entire network are shown. The user can choose to view data about individual hosts or vulnerabilities by selecting the corresponding button from the top of the pane. In Figure 4-6a, the item “bullwinkle” is selected from the Hosts list and its information is displayed below, including: its name, subnet, IP address, and asset value; the highest level of compromise achievable by the attacker; a breakdown of the specific vulnerabilities that exist on the host; and a list of its open ports. The vulnerabilities are separated by whether or not they can be exploited by the attacker, and the names are hyperlinks that jump to the corresponding item in the Vulnerabilities list. For example, clicking the “CVE-2001-0876” item will toggle the list view and display information for that vulnerability, visible in Figure 4-6b. The vulnerability listing contains: its identifier, its locality and effect (e.g. *Remote to Root*), a description, and the set of host ports affected by this vulnerability.

Network Map mode enables a system administrator to understand the general connectivity between subnets. It also allows administrators to explore the overall network model within the context of an attacker threat, and they are able to drill down into the network to view attributes of a vulnerability or identify the ways in which an individual host is vulnerable to attack. This information is provided as an initial step to understanding the security of a network.

### 4.3.2 Attack Graph

GARNET’s Attack Graph mode presents the same visualization of the attack graph as the Network Map mode, and it provides an interface for direct interaction with NetSPA’s network model. These interactions allow the user to perform “what-if” experiments by simulating the effects of hypothetical attack scenarios and changes

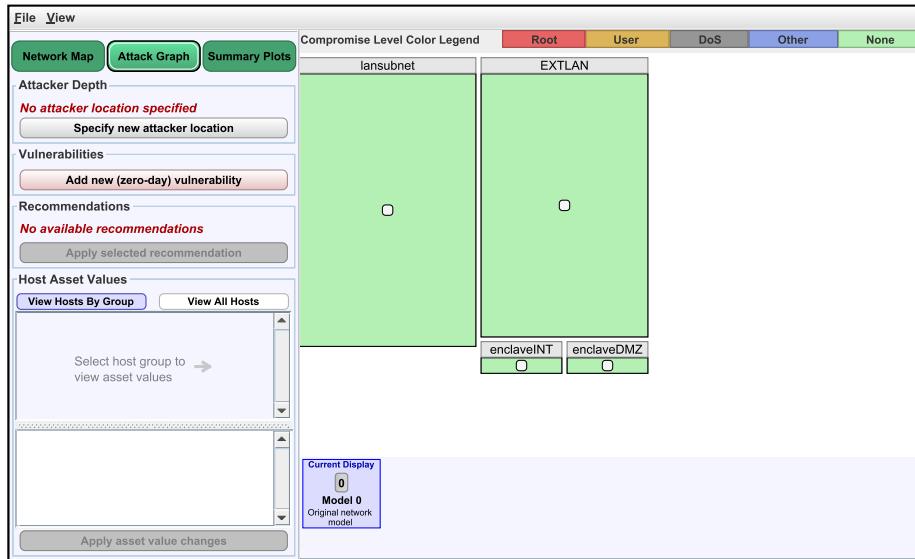
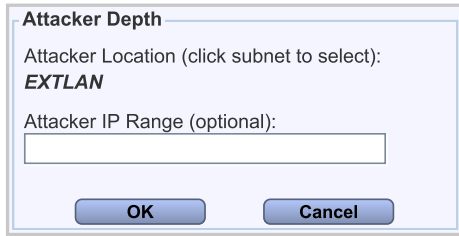


Figure 4-7: GARNET in Attack Graph mode.

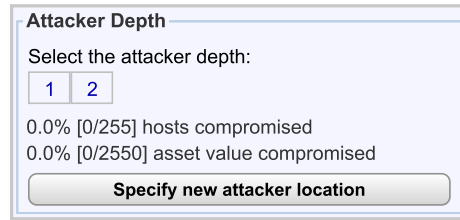
to the network configuration. The control panel for this mode contains four sets of controls for manipulating the display.

## Attacker Depth

When a network model is loaded into the tool, the attack graph is not initially generated and, consequently, all of the subnets appear as light-green, indicating no compromise. Figure 4-7 shows the Attack Graph mode interface in this initial state. The “Attacker Depth” pane lets the user specify the attacker’s starting location, or entry point into the network. Clicking the “Specify new attacker location” button causes a different set of controls to appear for specifying the relevant parameters (Figure 4-8a). The attacker location is defined by a subnet and an optional range of IP addresses. NetSPA, by default, builds its attack graph under the assumption that an attack is able to originate from any source IP address (this provides a good model of an attack that can come from anywhere on the Internet). The range of potential source addresses can optionally be constrained to model a more limited adversary or environments where IP spoofing is restricted. Once a starting subnet has been selected, the attack graph is built and the visualization is updated to reflect the resulting host groupings. The “Attacker Depth” pane is also updated, as shown in Figure 4-8b.

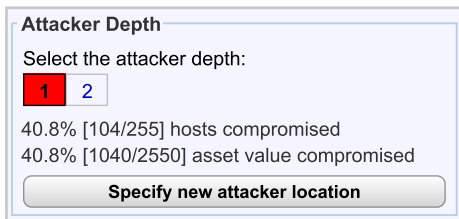


(a)

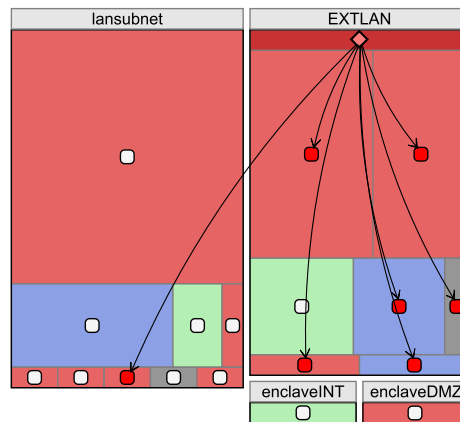


(b)

Figure 4-8: Views of the Attacker Depth pane after the “Specify new attacker location” button has been clicked (a), and after the attack graph has been constructed with the specified location (b).

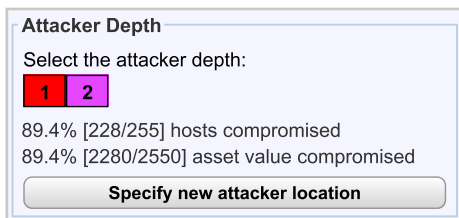


(a)

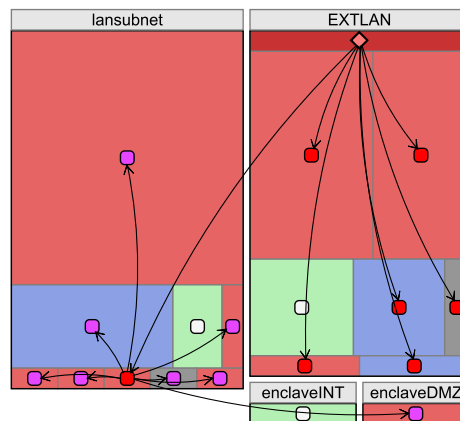


(b)

Figure 4-9: A depth of 1 is indicated on the depth control (a), and the edges for 1 attacker hop are drawn in the visualization (b).



(a)



(b)

Figure 4-10: A depth of 2 is indicated on the depth control (a), and the edges for 2 attacker hops are drawn in the visualization (b).

In the visualization, the edges of the attack graph are displayed incrementally through the use of the depth control, which consists of a set of adjacent numbered squares. The control in Figure 4-8b signifies that the attack graph is two levels deep. For the first attacker hop, the first level of depth is selected in the depth control (Figure 4-9a) and edges are drawn from the attacker node to all groups of vulnerable hosts that are directly reachable from the attacker’s initial location. Figure 4-9b illustrates this first hop for the sample network. The attacker begins on the “EXTLAN” subnet and, on the first hop, is able to compromise most of the hosts in that subnet, as well as a single group of hosts in “lansubnet”. Selecting the remaining depth number from the control (Figure 4-10a) draws edges from the host groups compromised at the first level of depth to the next set of compromisable hosts. Figure 4-10b shows that the second attacker hop compromises the majority of “lansubnet” and an additional host group in “enclaveDMZ”. As the edges are revealed, the target nodes become colored according to their level of depth so that they remain easy to distinguish as more links are drawn. In the example displays, nodes representing hosts compromised at one hop are colored red and nodes compromised at two hops are colored purple. These colors are also reflected in the depth control, and as subsequent depths are selected, the numbered boxes are filled with the appropriate color. In addition, the text below the depth indicator indicates the percentage of hosts and network asset value that has been compromised at the current depth.

## **Recommendations**

The set of controls in the “Recommendations” pane allows users to explore the automated recommendations generated by NetSPA and simulate their impact on the network. For each recommendation, the pane informs the user of the vulnerabilities that must be patched on certain host ports in order to protect a given portion of the network. It also indicates the total asset value and host groups from the attack graph that would be protected. The recommendations are ordered by the total asset value they protect, and the most effective ones are presented first. Users can see the effect of following a given recommendation by clicking the “Apply selected recom-

mentation” button. This actions causes GARNET to generate a new network model with the indicated vulnerabilities removed from the given set of host ports, and then rebuild the attack graph. In practice, this takes only a few seconds even for networks with thousands of hosts. During this process, NetSPA attempts to generate another set of recommendations, which could then be applied to the new network model to further protect it. An empty set of recommendations indicates that the removal of any additional vulnerabilities will have no effect on the attack graph, but it does not imply that the network is entirely secure. Some residual vulnerability will usually remain so that the standard network services are available and can operate normally.

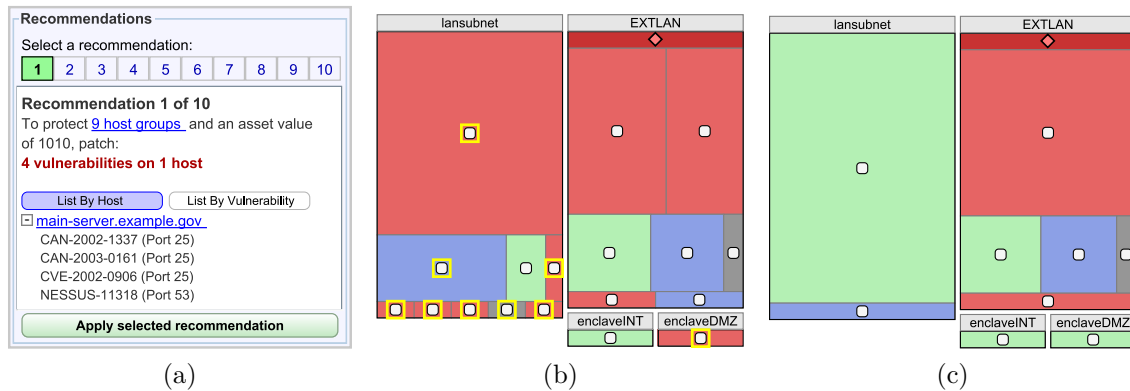


Figure 4-11: The first recommendation is selected in the Recommendations Pane (a), the host groups that would be protected are highlighted in the visualization (b), and the view of the attack graph is regenerated after applying the recommendation (c).

Figure 4-11a shows the first of ten recommendations for the sample network. The control at the top of the pane functions as a selector for the recommendations, and it conveys both the total number of items in the list and the currently selected item. The top recommendation indicates that four vulnerabilities must be patched on the single host “main-server.example.gov” in order to protect nine of the host groups represented in the attack graph. As illustrated in Figure 4-11b, the corresponding nodes in the display can be highlighted by clicking the hyperlink contained in the text indicating the number of host groups protected (in this case, it is the text “9 host groups”). Applying the recommendation yields the display in Figure 4-11c. If the updated display is compared with the previous one, it is immediately clear that the network is much less vulnerable to attack, since the subnets are now mostly green.

Additionally, the only hosts that can be accessed at a root level are in the attacker’s starting subnet.

## Vulnerabilities

In addition to viewing the effects of removing vulnerabilities by applying recommendations, a user can introduce vulnerabilities into the network. This functionality makes it possible to model different types of adversaries, as discussed in Section 3.2, and experiment with hypothetical attack scenarios. NetSPA’s default attacker model is the *simple attacker*; this has been the model used for the previous attack graph examples. Adding vulnerabilities to the network extends this model to more skilled attackers who use zero-day exploits.

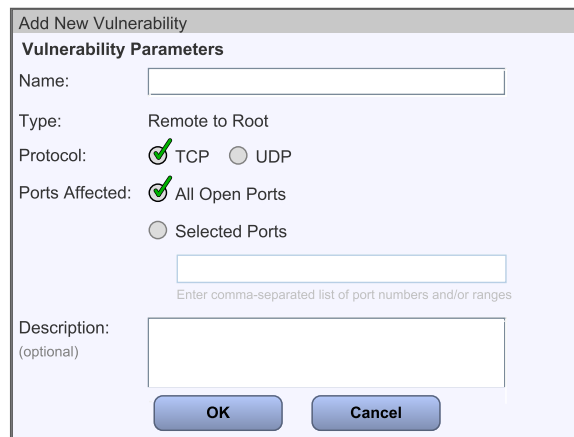


Figure 4-12: Dialog allowing a user to add a zero-day vulnerability.

The “Add Zero-day Vulnerability” button displays an input dialog for specifying parameters of the vulnerability to be added. The dialog, pictured in Figure 4-12, is used to assign a name, protocol, and an optional description to the vulnerability. The locality and effect of the vulnerability are fixed as *Remote to Root*. The set of affected ports must also be specified, and the dialog provides two options:

- *All Open Ports* will add the vulnerability to all open host ports in the network. This option models the *comprehensive-zero-day attacker* that has exploits for all conceivable zero-day vulnerabilities and can compromise the network to the fullest extent.

- *Selected Ports* allows a range of port numbers to be entered. The vulnerability will be added to just those hosts that have open ports matching any of the specified port numbers. This option can be used to model one to several *single-zero-day attackers*, who have the ability to exploit one zero-day vulnerability.

Once the parameters have been entered and validated, GARNET recreates the network model and attack graph with the vulnerability added to the specified ports and then refreshes the display with the new model. Figure 4-13a contains the attack graph representation of the sample network after the addition of a single zero-day vulnerability on one port. For the display in Figure 4-13b, a zero-day vulnerability was added to all open ports. Comparing these attack graph displays with the previous examples in Figures 4-9 and 4-10 shows the relative impact of each of these attacker models.

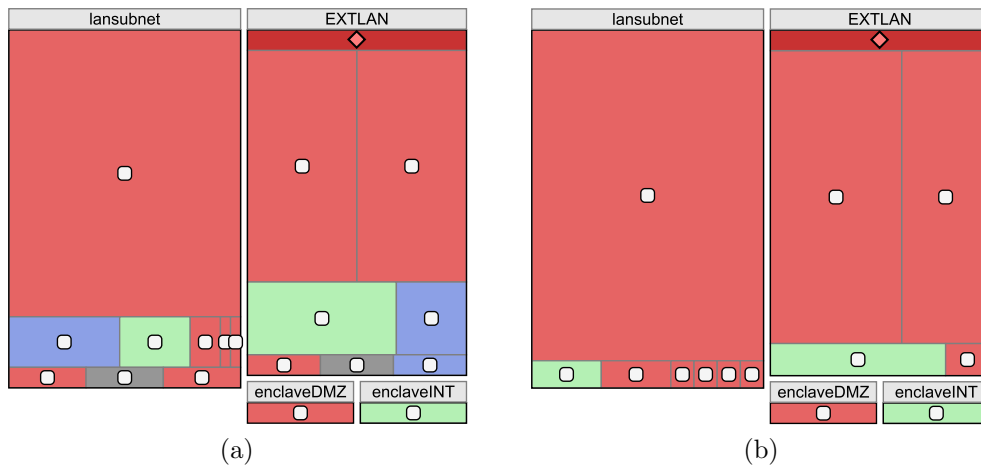


Figure 4-13: Visualization of sample network for a single zero-day attacker model (a) and a comprehensive zero-day attacker model (b).

## Host Asset Values

GARNET also allows the asset values for individual hosts to be modified. Since these quantities are used to prioritize recommendations and compute the security metrics, they can have a measurable effect on the steps an administrator should take to secure a network. For example, critical components such as servers and machines holding confidential information should clearly be treated as more valuable than other hosts.

The set of controls in the “Host Asset Values” pane is depicted in Figure 4-14a. A list of hosts and their current asset values appears in the upper section. This list can either contain all hosts in the network or just the hosts contained in a subnet or host group that has been selected from the display. The asset values are modified by clicking on the “plus-minus” icon next to the existing value, at which point the field becomes editable. Values that have been changed are colored according to whether the value has increased (blue) or decreased (red). A corresponding entry is also added to the lower section, as shown in Figure 4-14b. Changes can be undone by clicking the “X” icon next to the entries in the lower list. Once all of the desired values have been set, the changes are committed by clicking the “Apply asset value changes” button. The display is then updated according to the amended network model.

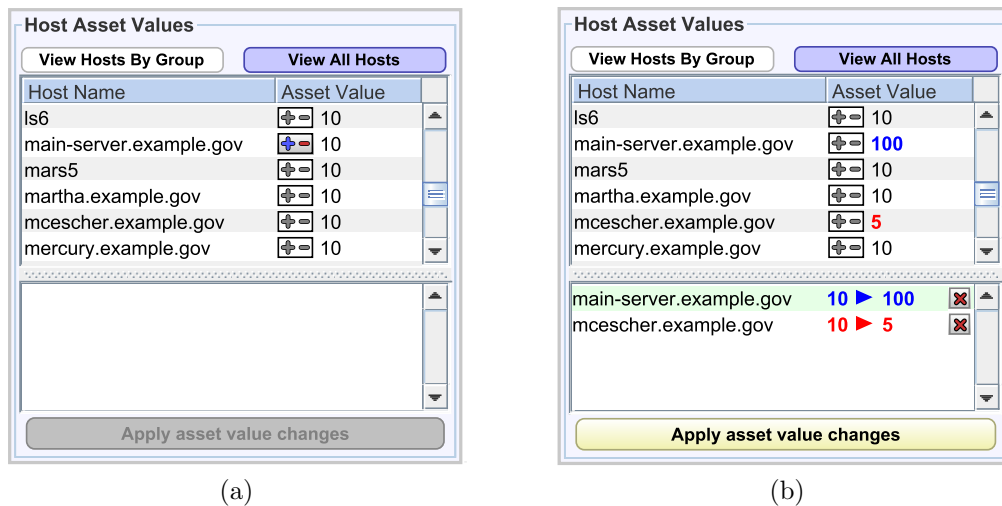


Figure 4-14: View of the Host Asset Values pane with no changes specified (a), and with two modified asset values (b).

### 4.3.3 Summary Plots

The third and final mode of interaction in GARNET enables users to view and compare NetSPA’s security metrics and also to enumerate vulnerability types in the network. In this mode, one of four different plot types can be chosen from the side panel, and the corresponding graph appears in the display area. Figure 4-15 shows the Summary Plots interface. The first three plot types correspond to the security



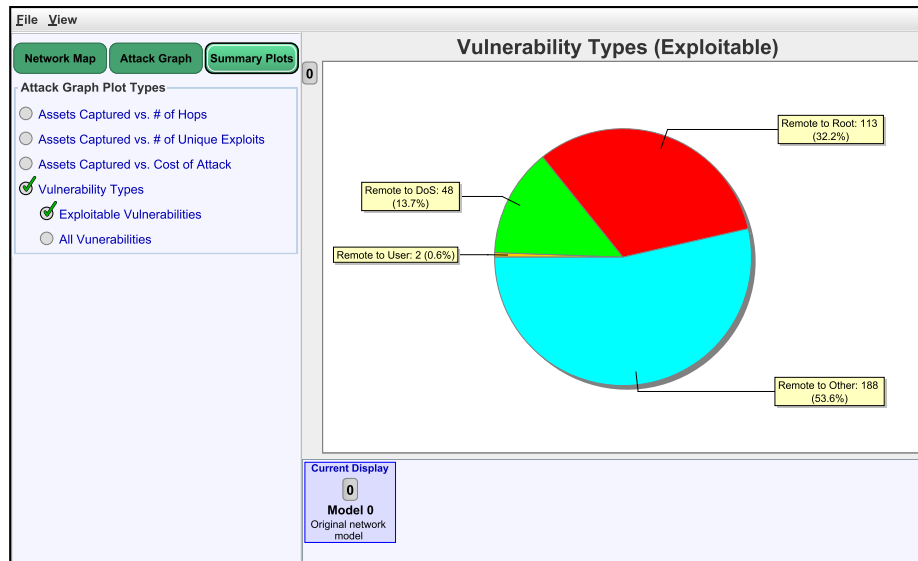


Figure 4-15: GARNET in Summary Plots mode. The vulnerability types chart for exploitable vulnerabilities is displayed.

metrics explained in Section 3.1.4, and their graphical representations are described below:

- **Assets Captured vs. Number of Hops** is displayed as a bar graph. Each bar corresponds to a single hop, or level of depth, and its height indicates the cumulative percentage of the network asset value that has been compromised at that depth.
- **Assets Captured vs. Attack Complexity** is represented as a stepped area chart, similar to a bar graph with no space between adjacent bars. The x-axis measures the cumulative complexity of the vulnerabilities (as defined by the CVSS [12]) that must be exploited to compromise groups of hosts in the attack graph. The height of the filled area corresponds to the cumulative percentage of assets that have been captured for a given complexity value.
- **Assets Captured vs. Number of Unique Exploits** is a line graph that contains multiple random curves, drawn as thin black lines, that indicate the percentage of network assets captured for each random unique exploit used. A bold red line is drawn along the upper bounds of the set of random curves to

approximate the ideal performance of an attacker using a set of unique exploits.

The fourth plot type is a pie chart of vulnerability types. Here, a *type* refers to a combination of locality and effect, e.g. *Remote to DoS*. There are two types of locality (*Remote* and *Local*) and four effects (*Root*, *User*, *DoS*, and *Other*), for a total of eight unique types. The user can choose to view a chart enumerating all vulnerabilities in the network or a chart encompassing only the exploitable vulnerabilities. The former is useful for a one-time overview of the network’s vulnerabilities, while the latter is more informative with respect to how the security of the network evolves as changes are applied to it. The pie chart for exploitable vulnerabilities in the sample network is displayed in Figure 4-15.

As users experiment with different network configurations and attacker models, these security metrics nicely summarize certain aspects of the attack graph and effectively reveal the degree to which the user’s changes impact the overall network security. Figure 4-16 shows the security metric plots for the sample network, before any modifications have been made. The graphs in Figure 4-17 illustrate the effect of applying a recommendation, which has protected nearly forty percent of the network’s assets from compromise. This view changes significantly in the face of a more capable attacker, and Figure 4-18 shows the computed metrics after the subsequent addition of a zero-day vulnerability to all open ports.

## 4.4 Timeline

As mentioned in the previous section, the Attack Graph mode enables users to experiment with “what-if” scenarios by modifying a particular aspect of the network. The network model and attack graph are regenerated and the display is updated accordingly. The actions that trigger the generation of a new network model include: applying a recommendation, adding a zero-day vulnerability, and adjusting host asset values. Several instances of network models can be created this way, and the interface uses a timeline component to visualize and manage the progression of models that are produced as a consequence of a user’s incremental modifications. The new models

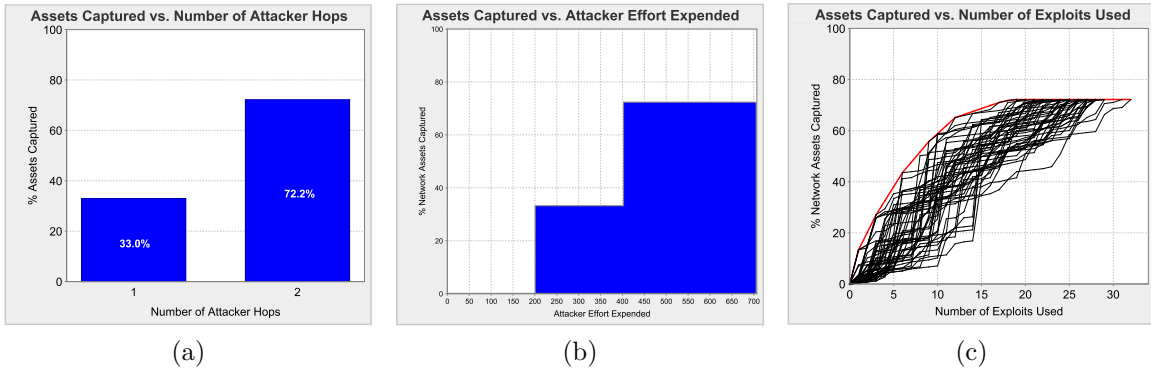


Figure 4-16: Security metric plots for the unmodified sample network.

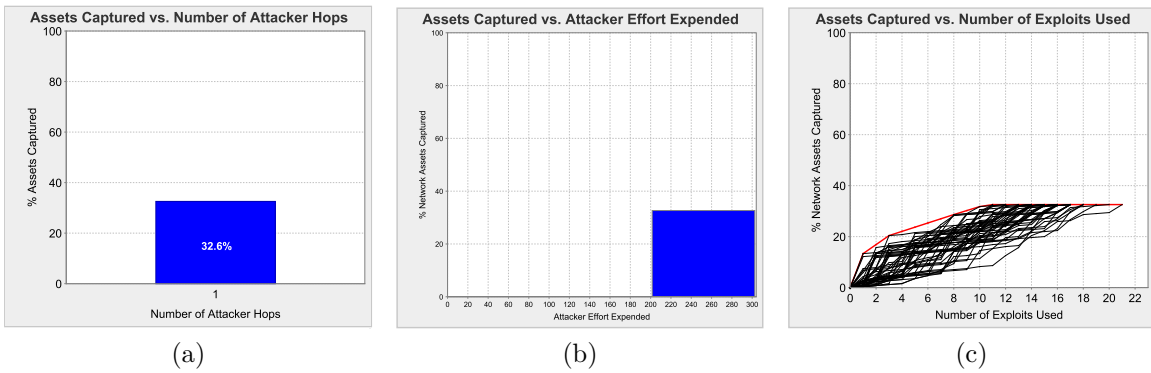


Figure 4-17: Security metric plots for the sample network after applying a recommendation.

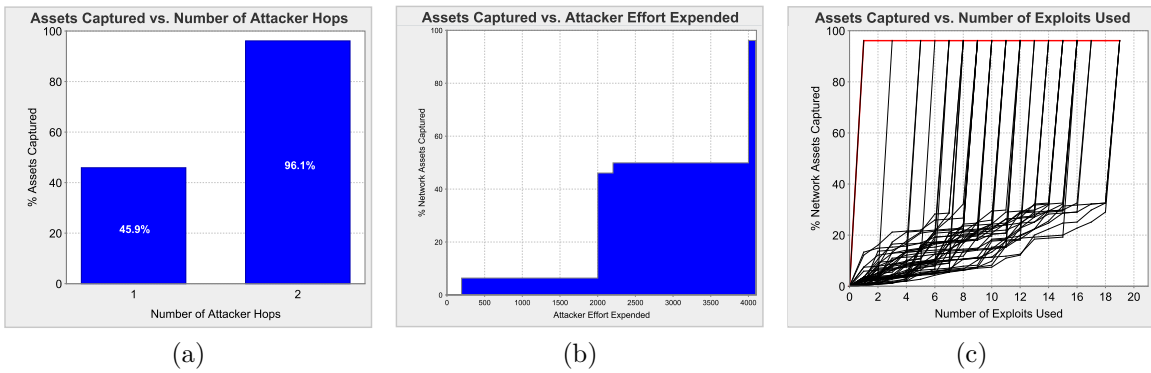


Figure 4-18: Security metric plots for the sample network after adding a zero-day vulnerability to all ports.

that are created represent the cumulative effect of all previous changes that have been applied, and the timeline enforces the notion that each subsequently generated model is a different version of the previous one.

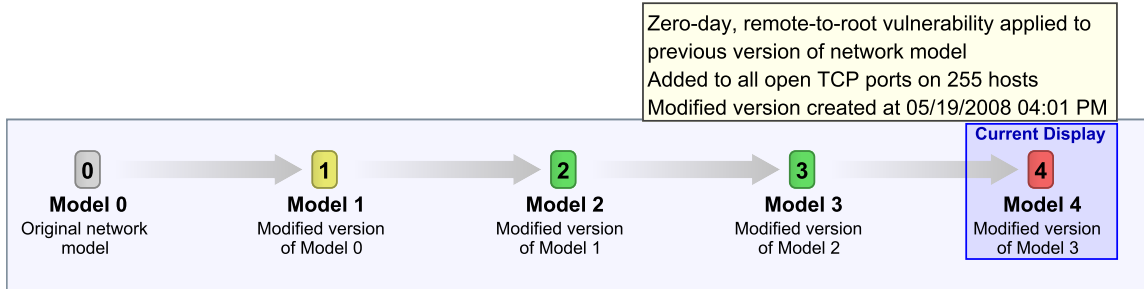


Figure 4-19: View of the timeline containing five versions of the network model. “Model 4” is selected and its tooltip is displayed.

The timeline is represented as a bar at the bottom of the interface, as shown in Figure 4-1. The component ties together the three modes of operation, and changes made in one mode are reflected in the other two. For each network version that is created, an icon, label, and short description is added to the timeline, beginning with the original network model that was loaded from disk. Each item also has an associated tooltip that provides a more detailed description of the changes that were added to the particular version. Figure 4-19 presents a view of the timeline containing five distinct network versions. “Model 4” is the currently selected model and its tooltip is displayed, indicating that it was generated as a result of adding a zero-day vulnerability to all open TCP ports in the network. Aside from the tooltips, a visual indication of the type of action that produced a given version is provided by the color of its icon (Table 4.2).

Icon Color	Type of action indicated
Green	A recommendation was applied.
Red	A zero-day vulnerability was added.
Yellow	One or more host asset values were modified.

Table 4.2: Mappings of icon color to type of action indicated.

To switch between the versions represented on the timeline, the corresponding icon can be clicked. This action retrieves the desired network model, refreshes the

display, and updates the information in the control panel. In Summary Plots mode, the timeline behavior is extended to allow multiple selection. Up to four items can be selected, and the corresponding charts are displayed side-by-side. This feature enables users to easily compare the summarized data for different versions of the network.



# Chapter 5

## Implementation

GARNET is implemented entirely in Java, using Swing as the graphical widget toolkit, and it is packaged as a JAR file. The application depends on the NetSPA shared library to run, as well as various third-party libraries. GARNET's implementation follows the MVC (Model-View-Controller) design pattern, and it is structured so that the model component provides a façade to the NetSPA interface, hiding it from the rest of the application. This design would allow additional views to be developed and plugged into the tool with relative ease.

The module dependency diagram in Figure 5-1 shows the overall architecture of GARNET. The following sections explain the system's interface to NetSPA and describe each of the major MVC components.

### 5.1 Interface to NetSPA

GARNET communicates with NetSPA through an interface that is generated by the SWIG toolkit. SWIG creates wrapper methods for a special C-based bridge to NetSPA's C++ code, then compiles it into a shared library. GARNET loads this library and uses the generated interface methods to invoke the C++ functions. The NetSPA API exports functionality for importing and parsing the binary files created by the importer and producing a network model. Through this API, three objects are generated that encapsulate the state of the network and its attack graph.

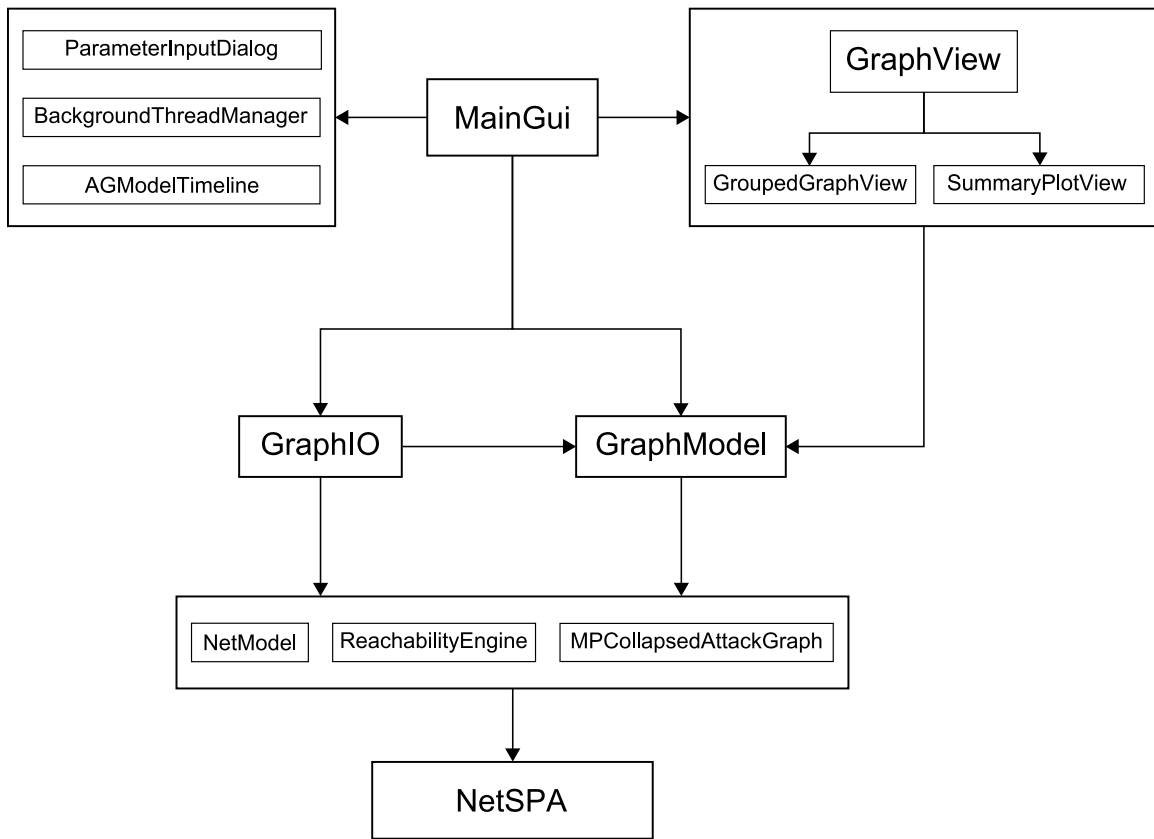


Figure 5-1: Module dependency diagram of GARNET.

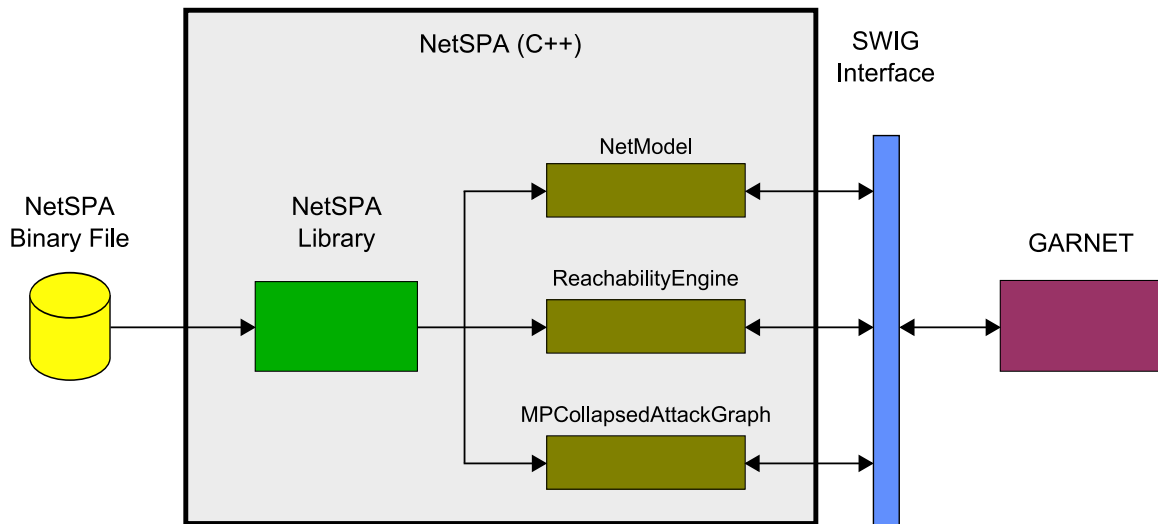


Figure 5-2: Block diagram of interface between NetSPA and GARNET.



These objects are: `NetModel`, `ReachabilityEngine`, and `MPCollapsedAttackGraph`. Figure 5-2 illustrates GARNET's interface to NetSPA.

NetSPA's network model is a collection of objects that represent characteristics and physical elements of a network – their relationship to each other is illustrated in the class diagram in Figure 5-3. A `Host` object represents a single computer in the network with a given asset value. A `Host` is associated with some number of `Intf` objects, symbolizing the interfaces through which a host connects to a part of the network. Each interface has its own IP address and is attached to a `Link`; a group of interfaces that connect to the same `Link` corresponds to a subnet. Furthermore, interfaces may have zero or more `Port` objects associated with them, representing a port on which the host can accept connections from other hosts. In general, a host will possess a single interface, and therefore is connected to a single network link, or subnet. Hosts with multiple interfaces that connect to different links are used to model firewalls and routers. In this case, the `RedirectRule`, `FilteringRule`, and `RoutingRule` objects are used to specify the rules according to which traffic can flow between subnets. The model also contains a set of `Vuln` objects which encapsulate data about network vulnerabilities, including a description, their locality and effect, whether there is an available patch, and measures of effort and complexity involved for them to be exploited. A vulnerability can be present on any port and a port can have any number of vulnerabilities. The `VulnerabilityInstance` objects create these associations and connect a `Vuln` to a given `Port`.

The network model created by the importer is compacted into a read-only data structure that is space-efficient and optimized for speed. To achieve this, all ID spaces are reduced to contiguous integer sequences. For example, if there are ten hosts in a network, the `Host` objects will have IDs spanning the range 0-9, with no gaps. The resulting structure is very simple, fast, and compact. The `NetModel` object is an abstract base class that provides an interface for accessing the information from the model. It has two direct subclasses: `StaticNetModel` and `DeltaNetModel`. A `StaticNetModel` is instantiated with a binary file containing the network model and simply allows read-only access to the underlying data structure. The `DeltaNet-`

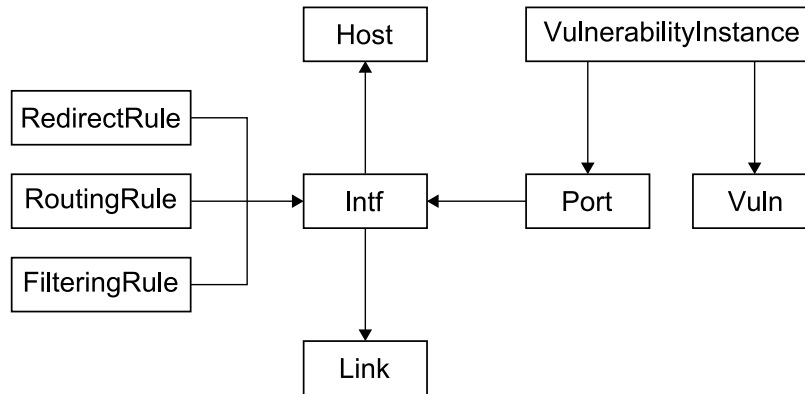


Figure 5-3: Block diagram of NetSPA’s network model objects. The links indicate many-to-one mappings, e.g. a **Host** may have many **Intfs**, but an **Intf** can only be associated with one **Host**.

**Model** is created from its static counterpart and enables modification of the network model. It supports the addition and deletion of objects, the modification of object properties, and the alteration of object mappings (e.g. the interfaces associated with a particular host, or the vulnerabilities present on a given port). The model itself is not altered; the changes are merely recorded so that queries to the altered model can be intercepted and the appropriate responses returned. After all desired changes have been made to a **DeltaNetModel**, they must be “committed” via a process that compacts the ID space, making sure that ID ranges are sets of contiguous integers. Once a **DeltaNetModel** has been compacted, it can be used in an identical fashion as a **StaticNetModel**, and it cannot be further modified.

The **ReachabilityEngine** computes host-to-host reachability within a specific **NetModel**. It provides methods for obtaining a list of ports that are reachable from any ordinary host or any attacker starting location. This component can also perform simple path traces and generate a textual description of the path a packet would follow if it originated at a particular link, passed through some combination of routers and firewalls, and finally arrived at a target port. Currently, the path trace function is limited to computing direct paths and it will only work for target ports that are directly reachable from the source link.

The attack graph and related data structures are encapsulated in the **MPCol-lapsedAttackGraph** object. It is initialized with a **ReachabilityEngine** and a **Link**

ID indicating the starting location for an attacker, then creates a set of interconnected node objects that mirror the structure of a collapsed multiple-prerequisite graph. The `MPCollapsedAttackGraph` is also responsible for generating recommendations and security metrics, as `Recommendation` and `MetricSeries` objects, respectively. Each `Recommendation` identifies the particular `VulnerabilityInstances` to be patched, as well as the hosts and combined asset value that would be protected as a result. For the three security metrics that NetSPA computes, there are three corresponding methods, each of which produces a `MetricSeries` object. The `MetricSeries` serves as a simple container that holds a set of x-y coordinates for the data to be plotted.

## 5.2 GraphModel

GARNET maintains its own internal representation of the network model, apart from that provided by NetSPA's interface objects. Since it is relatively expensive for the Java code to call into the C++ library, the application extracts and precomputes a significant portion of the information from the `NetModel`, `ReachabilityEngine`, and `MPCollapsedAttackGraph` objects. This task is performed by the `GraphIO` class and the extracted information is stored in a `GraphModel`. The `GraphModel` object encapsulates a directed graph abstraction (implemented using the JUNG library [7]) with annotated nodes and edges. The nodes in the graph correspond to the collapsed state nodes from an `MPCollapsedAttackGraph`, representing a group of `Host` objects that have identical reachability as determined by the `ReachabilityEngine` and that can be compromised at a particular level of access in the attack graph. Each node is associated with a name and ID, as well as its depth in the attack graph, the level of access achievable by an attacker, the subnet to which its hosts belong, and a list of IDs identifying the set of `Host` objects it represents. The edges between nodes are labeled with the type(s) of relationships they represent. These relationships include an attack path, indicating that the set of hosts in the target node contain vulnerabilities that an attacker can exploit from the set of hosts in the source node, and a reachability path, indicating that the set of hosts in the source node can connect to ports on those in the

target node. Using this graph abstraction and given a particular host or node ID, the `GraphModel` can rapidly respond to queries for access levels, depth, reachable hosts, and attacker hops. The `GraphModel` also provides an interface for accessing host-specific information (e.g. names, IP addresses, and asset values), recommendations, and security metric data. However, this information is not locally stored in the object, and it must be retrieved through the C++ library.

In addition to providing an interface to the NetSPA data structures, the `GraphModel` has the ability to produce modified copies of itself based on requested changes to the underlying network model and attack graph. It declares methods that allow the attacker location to be altered, recommendations to be applied, vulnerabilities to be added, and asset values to be updated. Changing the starting location of the attacker only requires the generation of a new `MPCollapsedAttackGraph` from the existing `NetModel` and `ReachabilityEngine` objects. For the remaining functions, a `DeltaNetModel` must be created from the original `StaticNetModel`. Once the changes have been applied, new instances of the `ReachabilityEngine` and `MPCollapsedAttackGraph` are generated.

## 5.3 GraphView

The visualization of the information within a network model is handled by the `GraphView` class. Instances of this view object are associated with a single `GraphModel` and provide an interface for manipulating the display. There are two manifestations of `GraphView`, each of which presents a different aspect of the model: `GroupedGraphView` and `SummaryPlotView`.

The `GroupedGraphView` is responsible for creating an interactive view that separates the graph nodes into visually distinct groups. This display is used for the Network Map and Attack Graph modes described in the previous chapter. The same `GroupedGraphView` object is used for the display in both of these modes and, as a result, the object itself has two corresponding modes in which it shows only reachability links or only attack paths. When the object is toggled between the two modes,

it saves its current state (i.e. which links have been made visible), and this state is restored the next time the modes are switched. An open-source toolkit called Piccolo [21] provides all of the drawing functionality for `GroupedGraphView`. It performs efficient rendering of 2D graphics and has built-in support for handling animation and direct interaction with visual objects. The Piccolo framework is based on a hierarchy of objects, where all displayed objects are children of a global root element. This root object serves as the canvas on which every object is painted. The individual sub-net groups in the display are composite visual objects, consisting of `NetworkGroups`, `NetworkSubgroups`, and `NetworkNodes`. Each group is embodied by a `NetworkGroup` object which functions as a container for a label bar and a collection of `NetworkSubgroups`. A `NetworkGroup` produces the treemap layout for its subcomponents and also listens to click and drag events from the mouse that indicate selection, resizing, or repositioning of the group. `NetworkSubgroups` keep track of their bounding boxes within their parent group and paint themselves with a color representing the appropriate compromise level. These objects, in turn, contain a single child `NetworkNode`, which is centered within its parent's bounds. Each `NetworkNode` displays itself as a small rounded shape and maintains its own tooltip object. `NetworkNodes` also respond to mouse events, and they display their tooltip or highlight themselves, as appropriate. The links drawn between individual nodes are implemented by `ArrowNodes`. They are added as direct children of the root canvas, and each has a source and target `NetworkNode`. The `ArrowNodes` draw themselves between these two points as a Bezier curve, which is recomputed as the positions of the associated nodes change.

The second type of view is the `SummaryPlotView`, which implements the display for the security metric plots. It employs the JFreeChart library [5] for its chart-rendering functionality. As previously mentioned, this view is capable of generating four different types of charts: a bar graph that illustrates the percentage of assets captured versus the number of attacker hops; a stepped-area chart depicting assets captured versus attack complexity; a line chart showing random curves for the assets captured as a function of the number of unique exploits used; and a pie chart for types

of vulnerabilities. When GARNET is in Summary Plots mode and several network models are simultaneously selected from the timeline, there can be several `SummaryPlotViews` displayed on the screen at once. Each instance must query its `GraphModel` for the chart data and this can be a costly operation, especially for the unique exploit plot where multiple curves must be computed on the fly. Consequently, the chart objects themselves are cached and stored within their corresponding model object so that the time penalty is only incurred once for each unique network model.

## 5.4 MainGui Controller

The `MainGui` class is the primary controller object for GARNET and it coordinates all interactions between the model, view, and timeline components. This class instantiates the main menu and all of the side panel controls as child components, listens for actions performed on them, and triggers the appropriate events. In particular, it handles the following main tasks:

- **Obtaining the `GraphModels` used to produce the displays.** When a file is initially loaded into the tool through the file menu, `MainGui` uses the `GraphIO` class to read in the network model through NetSPA and produce a corresponding `GraphModel` object. Similarly, when the model is changed by specifying a new attacker location, adding or removing vulnerabilities, or modifying asset values, the appropriate methods are invoked to create a new `GraphModel` based on the current one. Once these model objects have been created, `MainGui` forwards them to its view components, which update themselves to reflect the new model.
- **Displaying the appropriate view when the mode is changed.** Switching between the three modes causes the corresponding view component to be displayed, and `MainGui` sets the visibility of the components accordingly. As mentioned in the previous section, a single `GroupedGraphView` is used for two of the modes, so this view is only hidden when the Summary Plots mode is invoked. Conversely, the `SummaryPlotView` is hidden upon entering the Network

Map or Attack Graph mode.

- **Updating the displayed view in response to the side panel and timeline controls.** For each of the interaction modes, the side panel contains a set of controls that allow manipulation of the currently displayed view. These controls send events to the controller, which makes the appropriate method calls to the view objects. This set of actions includes: showing/hiding reachability links, incrementing/decrementing the attack graph depth, and selecting a plot type to display. The timeline control, on the other hand, affects all views simultaneously. It maintains a sequence of `GraphModels` that is updated each time `MainGui` obtains a newly generated model. Whenever a model version is selected from the timeline, `MainGui` retrieves the associated `GraphModel` and essentially performs the same tasks as it does when a model is created, including updating the views and repopulating the controls.
- **Generating dialogs for user input.** `MainGui` is responsible for showing the popup dialogs that appear for choosing a layout and adding a zero-day vulnerability. Once the user has entered and accepted the desired parameters, the information is passed along to the component that is intended to handle the particular event.
- **Managing long-running tasks with background threads.** Although GARNET achieves rapid interaction with NetSPA, building the attack graph and chart displays are relatively expensive operations and, for large networks, may take more than a few seconds to complete. To handle these situations, `MainGui` spawns a separate background thread each time the displays must be reconstructed from a new `GraphModel`. Since these tasks are executed asynchronously, the application remains responsive during long-running operations. The controller does, however, disable interaction with the majority of the interface and displays a progress dialog while a background task is running.





# Chapter 6

## Evaluation

A user evaluation was performed on a previous iteration of the GARNET application in order to assess the effectiveness of its visual representation and the usability of its interface. The resulting feedback was used to refine the tool's design. The final version of GARNET, which is presented in the previous chapters, differs from the evaluated version only in terms of the interface design, and the functionality is essentially the same in both cases.

### 6.1 Testing Methodology

The evaluation of user interfaces is not generally a formalized process and it can be done in various ways. Researchers in the field of user interface design have suggested that formal methods, such as formulating a proper analysis model or applying a computerized procedure, are impractical for most applications [17]. As a result, a more informal technique known as heuristic evaluation was employed for GARNET's user testing. This method involves presenting evaluators with a user interface and asking them to subjectively judge the interface according to a set of usability guidelines, known as heuristics. The guidelines used for evaluating GARNET were drawn from a well-known set of ten heuristics developed by Jakob Nielsen [16].

A group of five users was recruited for this evaluation. All of the users were members of technical staff at the MIT Lincoln Laboratory, and they were all familiar

with attack graphs and network security concepts. Two of the users had been exposed to earlier iterations of the tool, while the remaining three had not previously seen the interface. Each person was given a brief overview of GARNET and the evaluation procedure (Appendix A), as well as the list of heuristic guidelines. The evaluators were encouraged to explore the interface as thoroughly as possible, using the guidelines to help them identify aspects of the tool that represented either a positive feature or a usability problem. They were also asked to rate each problem in terms of severity and to suggest a solution if possible. The participants were given approximately a week to complete their evaluations, which they did independently and at their own convenience.

## 6.2 Results

The evaluators each produced a list containing descriptions of strengths and weaknesses of the interface and recommendations for improvement. The number of comments ranged from 14 to 46 items, with a median of 44. These evaluations are included, in their entirety, in Appendix B.

In general, the users thought the interface was aesthetically pleasing, and perceived the layout to be clean and simple. They liked the compactness of the treemap representation and the fact that the individual subnet rectangles could be directly manipulated. They also responded positively to the ability to simulate hypothetical changes and to the responsiveness of the system in creating and switching between the network models. Additionally, the users gave positive comments about the depth control, which enables stepping through the attack graph one attacker hop at a time and uses color to group the nodes in the display by depth.

The remaining comments that pointed to the drawbacks of the interface were condensed into a list of 55 distinct items. This summary, presented in Appendix C, was used as a guide to identify bugs in the implementation and improve upon the existing features. The items are divided into four categories based on the severity of the problem they represent, namely: Critical, Major, Minor, and Cosmetic. In

implementing GARNET’s final design, each of the three critical items and about half of the remaining items were addressed.

The first critical item involved the overall organization of the interface controls, and it represented a problem for all of the users. In the evaluated version of the interface, there was an additional set of toggle buttons at the bottom of the control panel for showing and hiding the Network Information pane, and it could be accessed from all three modes. Also, the Host Asset Values pane only appeared in the controls for Network Map mode. Figure 6-1a shows the original control panel layout for the first two modes. The majority of the evaluators found it difficult to locate information using this interface and thought some of the interactions were inconsistent across modes. To address these concerns, the side panel controls were reorganized so that the Network Information and Subnet Reachability panes appeared in Network Map mode, and the controls for all interactions that alter the network model were unified in Attack Graph mode. The two extra selector buttons were also removed from the control panel. The improved layout is displayed in Figure 6-1b.

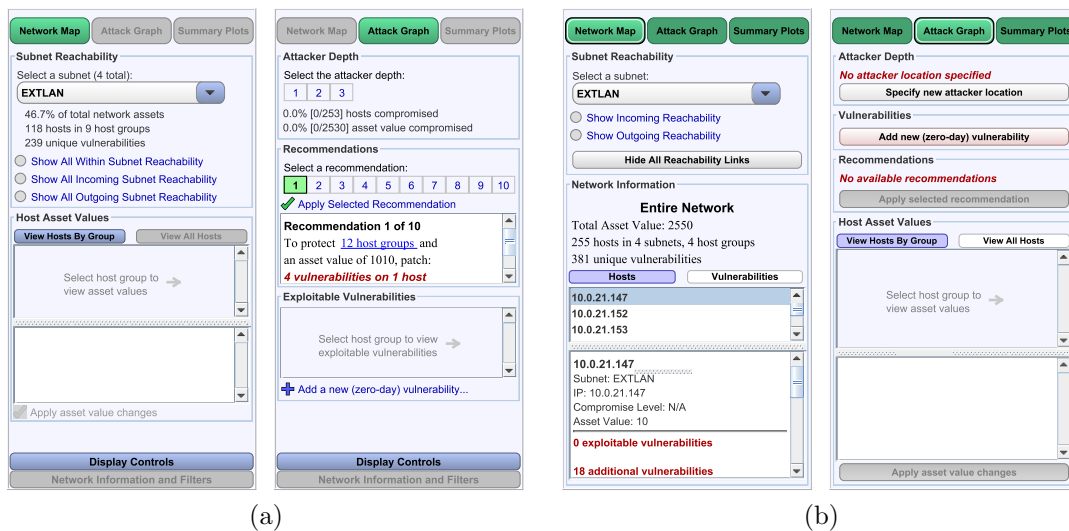


Figure 6-1: Original control panel layout for the Network Map and Attack Graph modes (a), and the improved layout (b).

Another problem concerned the implementation of the timeline. The original design, pictured in Figure 6-2, used small icons placed along a thin bar to represent the different network model versions, and the labels were short and not very descriptive.

In the first two modes, a set of arrows was used to toggle between the timeline versions, while direct selection of the icons was allowed in the third mode. All of the evaluators mentioned that the interaction with the timeline was inconsistent and nonintuitive. Figure 6-3 illustrates the redesigned component, which uses more descriptive icons and text. It also functions consistently across modes and uses better visual cues to indicate the selection and progression of network models.

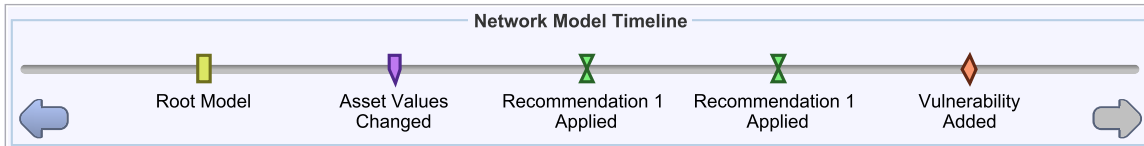


Figure 6-2: Previous design of the timeline component, containing five network versions.

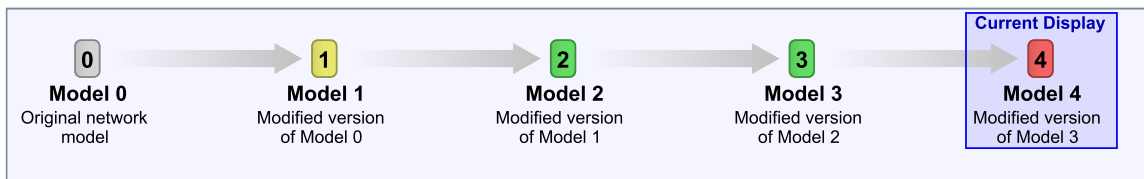


Figure 6-3: Improved design of the timeline component, containing five network versions.

The final critical issue came from comments about the interface for specifying the attacker starting location. Initially, an input dialog for these parameters was displayed immediately after loading a model from file. The interface required users to provide this information before creating any of the displays, and the evaluators found it difficult to determine the appropriate subnet for the attacker without being able to access any other information about the network. Furthermore, the location could only be specified once, and there was no way to change it once the model was loaded. The evaluations pointed out that it cannot be assumed that a user would be able to make this determination without specific knowledge of the particular network. This part of the interface was changed so that the tool loaded a model and produced the visual representation with no attacker location defined. It now allows users to first explore the network before choosing an attacker location and to later modify this selection at any point.

The GUI was altered in numerous other ways in response to the other items in the summarized list. Some of these additions included: adding a legend to the display area explaining the colors used for compromise level; supplementing the per-host information with a listing of the host's open ports; providing the automatic layout options; and using background threads and progress dialogs for extended computations. Various bugs were also discovered and fixed.



# Chapter 7

## Conclusion

### 7.1 Contributions

This thesis has motivated the need for the development of an effective attack graph visualization and it has presented the design and implementation the GARNET application. This tool produces a compact representation of an attack graph that visualizes the reachability within a network and the ways in which it can be compromised by an attacker. It provides an informative interface that enables access to information about individual hosts and the vulnerabilities they possess.

GARNET interfaces with an existing framework that automatically generates and analyzes attack graphs, and it presents recommendations that suggest preventative actions as well as metrics that summarize the overall security of a network. The interface enables users to experiment with hypothetical “what-if” scenarios by simulating the effects of changes to the network configuration. The supported actions include: following recommendations by removing vulnerabilities, introducing new zero-day vulnerabilities, and modifying host asset values. Further, the attack scenario can be changed by varying the initial attacker location.

An initial round of user testing produced a set of subjective evaluations for the graphical user interface. The evaluations were a valuable source of feedback about the usability of GARNET’s interface. They confirmed the tool’s effectiveness in conveying information about the attack graph and providing a set of interactions that allows

users to experiment with different scenarios. The recommendations about problematic areas of the interface aided in the development of a more functional design, while many of the comments pointed to larger issues that provide directions for future work.

## 7.2 Future Work

Although GARNET successfully conveys a significant amount of information through its visual representation, it is still somewhat limited in its illustration of overall network topology. The user is able to view reachability links between groups of hosts in different subnets; however, for numerous host groups and dozens of subnets, displaying this reachability all at once can produce a confusing jumble of edges. A potential alternative to displaying the individual links would be to utilize a flow map technique [20], resulting in merged edges whose varying widths indicate the number of inbound/outbound connections. Furthermore, the tree structure of the flow map could be used to dictate the initial layout of the subnet groups, and filtering devices (such as routers and firewalls) could be shown along the edges between the subnets they connect. This view would provide a clearer picture of the physical connectivity of the network.

Another possible extension would extend the interface to support additional methods of interaction with the network model, such as altering firewall rulesets and repartitioning the network. This would allow a larger set of “what-if” scenarios to be generated and enable system administrators to experiment with additional aspects of the network configuration. Additionally, support could be added for preserving the modified versions of the network model that appear on the timeline and writing them out to disk. Users could thereby save the state of their experimentation with a network and load the models back into the tool at a later time.

Further work could also expand the set of attacker models to include client-side attacks. For this category of exploits, an attacker could use a compromised server to attack vulnerable client machines or send malicious email attachments that would infect the target host.



Finally, GARNET should be exposed to further rounds of user testing, including empirical evaluation by system administrators. This form of user evaluation would involve presenting a set of target users with the interface and measuring how well they perform various tasks that focus on important aspects of the tool's functionality.



# Appendix A

## User Evaluation Briefing

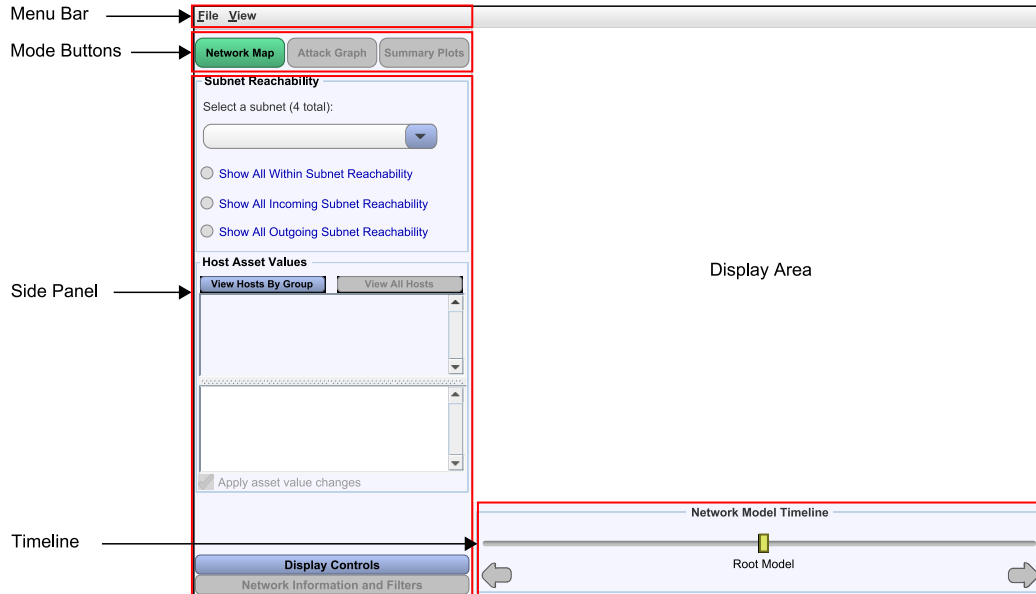
The briefing presented in this appendix was provided to all users who participated in the heuristic evaluation for GARNET.

### Introduction

This application functions as a visualization tool for network attack graph data, providing a simplified and intuitive understanding of key weaknesses discovered by attack graph analysis. It takes the output of an efficient C++ computation engine, a program named NetSPA that performs attack graph and reachability computations for a network model, and builds a visual representation of this information. Important features of the nodes in the graph are conveyed by grouping, size, and color, while edges and other attributes are initially hidden and can be displayed on demand. The tool provides a set of controls for exploring host-to-host reachability, discovering critical attack paths between groups of hosts, and viewing overall statistics about the network and the attack graph. In addition, it allows certain changes to be made to the network model, such as addition/removal of vulnerabilities and modification of host asset values, and dynamically regenerates the display to reflect the updated model.

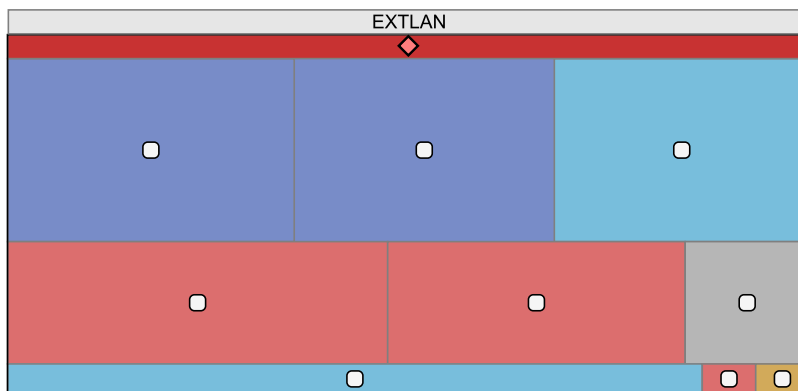
## Interface Overview

The user interface for this visualization tool consists of 4 main display and control components, plus a menu bar.









## Description of Visual Representation

In certain modes, a visual representation of the attack graph appears in the display area of the interface. The display is comprised of one to several labeled rectangular groups, which are further divided into subgroups of various colors. The figure below shows an example of one of these groups.



The approach used for this attack graph display only incorporates a subset of nodes from the graph, known as collapsed state nodes. A collapsed state node represents

a group of hosts that are treated identically by filtering devices in the network, and on which an attacker has obtained a particular level of access. In other words, the hosts in each of these groups are, from a security standpoint, essentially the same. These nodes are grouped according to the subnet to which their constituent hosts belong. Each node is symbolized by a white square, and appears within a colored subgroup of its corresponding subnet. The color of the nested groups indicates the level of attacker compromise for that group of hosts; the five possible levels of access are root, user, denial-of-service (DoS), other, and no access. A sixth color is used to distinguish the actual attacker (represented by a pink diamond). The subgroup colors are defined according to the following legend:

	(coral)	Root Access
	(orange)	User Access
	(cyan)	DoS Access
	(dark blue)	Other Access
	(gray)	No Access
	(deep red)	Attacker

The subgroups are proportionally sized according to the number of hosts contained in their corresponding state nodes.

## Evaluation Procedure

Here's an outline of the procedure that should be used for this evaluation:

- Inspect the interface as thoroughly as possible, and attempt to explore all of the functionality
- Identify aspects of the interface that represent either a positive feature or a usability problem, and provide a brief description
- Use the attached heuristic guidelines to categorize each problem found (but don't limit yourself just to those listed)

- Judge the severity of each problem using the following scale:
  - *Major* represents a significant lack of functionality or poor usability
  - *Minor* represents similar problems that do not severely impact functionality
  - *Cosmetic* refers to visual aspects that could be made to look more consistent or aesthetic, but do not affect the functionality of the system
- Suggest a solution, if possible

The entire process may last about 2-3 hours, and it should ideally be done in a single session (but it can be spread out over no more than 2 sessions). In order to keep the results unbiased, make sure to work independently. Finally, try to be as picky as possible – even if a problem seems insignificant, it’s probably worth noting.

### **Necessary Files**

A ZIP file with the executable JAR and all of the necessary libraries is provided. To start the application, simply unzip the files and run viztool.jar. The network models are included as binary (.bin) files, and these are the files that are to be loaded into the tool. There are 9 test networks included in the ZIP file – one is an anonymized version of the Group 62 network, and the others are smaller example networks that were created by hand. The specifications of each are attached.

Do this at your own convenience and email me about any points of confusion or other questions you have. Thank you.

# Appendix B

## User Evaluations

This appendix contains the five evaluations collected from the users who participated in the heuristic evaluation for GARNET. The original text was only slightly modified, and it was reformatted for clarity and edited for spelling.

### User 1

- Good: (Aesthetic) Nice Clean look
- Good: (Aesthetic) Separation of Model/View/Controller, Controls are on the left and bottom (mostly)
- Good: (Aesthetic, Learnability) Graph Interactions (dragging, etc) are intuitive
- Good: (Feedback) Contextual information (mouseover popups throughout)
- Major: (Aesthetic, Learnability, Consistency) grayed out buttons convey the wrong message — makes me think features are disabled, instead, selected items should look the same, but have some sort of highlight (e.g. border different color) like in other GUIs
- Major: (Learnability) it was initially unclear how the timeline comes into play at all with this tool — it actually took a while of use before I figured out how to influence the timeline and even then upon applying recommendations, it is unclear that the timelines are tied to the application of recommendations
- Major: (Aesthetic, Consistency) it is also not immediately clear which point on the timeline a user is on — the yellow and green and gray markers are confusing — why is the Root Model different? How come I can't just click on one to go to it? It looks like a scroller, but doesn't act like one.

- Minor: (Learnability) the tabs for Display Controls and Network Information Filters are non-obvious to the user since the other navigation buttons are up top, but they are down below
- Minor: (Learnability) It says to select a host group to view asset values, but it isn't immediately clear how to select a host group or that the boxes in the graph are the host groups
- Minor: (User Control and Freedom) default layouts are a pain to use in some cases — requires a lot of scrolling for networks with many subnets
- Minor: (User Control and Freedom) it is not necessarily clear what IP ranges are represented by the labeled subnets, if I were a network admin, the text labels might be ambiguous. There is also no visualization of physical/logical connectivity between the subnets
- Minor: (Consistency) why is the slider bar on the timeline highlighted when I select the “summary plots” button?
- Minor: (Consistency) there is no legend that says what colors map to (root, user, DoS, etc) in the graph, and when they are charted in the summary plots, the colors that are used to talk about the same things are not the same
- Minor: (User Control and Freedom) why just the top 10 recommendations, why not a list for all recommendations?

## User 2

- (Cosmetic - Aesthetic and minimalist design) Title bar has extraneous information like build number and date that belong to an About dialog, generally accessible through the menu bar.
- (Minor - Error prevention) It is good that buttons and menu options are disabled at startup, which narrows the number of possible actions the user could take to File->Load. But how about taking them straight to the Network Model file chooser when they start VizTool?
- (Cosmetic - Consistency and standards) View menu should be disabled at startup since all submenus are disabled.
- (Cosmetic - Aesthetic and minimalist design) Put a bit of space between “Display Controls” and “Network Information and Filters” for consistency and aesthetics.
- (Good - Aesthetic and minimalist design) The overall viewer is simple and aesthetically pleasing. You choose good colors and nice widget shapes. The viewer is small, which is good for people with low-resolution monitors.



- (Minor - Help and documentation) A Help menu should be included.
- (Minor - Flexibility and efficiency of use) Menu items should have have “Ctrl” shortcuts in addition to “Alt” mnemonics.
- (Cosmetic - Consistency and standards) I suggest flattening out the menu hierarchy. Traversing menus and submenus takes extra time. It’s more convenient to have “Save Layout” and “Save Display as Image” on the same level as the two Load options, for example.
- (Cosmetic - Match between system and the real world) In the Load Network Model dialog, the “Cancel” button’s tooltip reads “Abort file chooser dialog”, which is Java-speak. In fact, “Open” and “Close” don’t really need tooltips.
- (Minor - Flexibility and efficiency of use) The current directory of the Load Network Model dialog always starts out as the directory that viztool.jar is in. This could be annoying if the user stores all his data in another far-away directory. It would be nice to remember the last opened directory, at least within a session.
- (Good - Visibility of system status) Loading is pretty fast, and there’s progress bar to provide feedback.
- (Cosmetic - Aesthetic and minimalist design) In the Specify Attacker Parameters dialog, “OK” and “Cancel” buttons are not the same size.
- (Good - Error prevention) Good defaults in the Specify Attacker Parameters dialog. The user doesn’t even need to know what they mean.
- (Minor - Help users recognize, diagnose, and recover from errors) It would be helpful to see what value “Automatic” sets. Better yet, use that value as the default for “User defined”, so it would be fine even if they select that, and it gives a little more guidance in picking a custom value.
- (Good - Error prevention) The IP-checking-as-you-type feature is very nice.
- (Minor - Consistency and standards/Aesthetic and minimalist design) The Timeline takes quite a bit of screen real estate. I suggest shrinking the arrows ...or do you really need the arrows? I noticed that in some modes, the time ticks are clickable, and in others, you have to use the arrows. This is a consistency issue. Personally, I think clicking on the time ticks are more direct.
- (Good - User control and freedom) Timeline: the ability to jump back and forth between changes is a great feature! Good tooltip info.
- (Cosmetic - Match between system and the real world) The term “Root” model/version is a bit of a jargon. I would use “Initial” or something.

- (Minor - Consistency and standards) The timeline looks like a slider and made me want to drag it like a slider when I first saw it. I didn't know what to do with the Root Model handle until I started playing with some other features. I also wonder why the Root Model handle starts in the middle instead of the left end of the timeline.
- (Major - User control and freedom) Cannot save any changes made to the network model or the Network Model timeline.
- (Minor - ) In the display, if the group is small, it is almost entirely covered by the white node, which makes it hard to see what the group's color is.
- (Minor - Flexibility and efficiency of use) Mousing-over a white node gives great detail stats. Since a white node and the corresponding color region represent essentially the same information, how about popping up the details when the user mouse-over the region? Then she wouldn't have to hit the node, a small target, precisely. I also wonder if you need the white node at all if the node and the area represent the same thing?
- (Major - User control and freedom) Reset Layout does not undo some sequence of Auto-Size Layout, Vertical Layout, etc.
- (Cosmetic - Aesthetic and minimalist design) The title of the popup is "Host Group N", but N is just a meaningless number to the user.
- (Major - ) I don't understand the function of the four radio selections at the bottom of the popup. They don't do anything. Sometimes they appear checked, but I don't understand why.
- (Major - Visibility of system status) Buttons look the same when they are disabled and when they are enabled but not selected. Since gray generally implies disabled, perhaps white would be a better choice for not selected?
- (Cosmetic - Aesthetics and minimalist design) In Network Map mode, "Select a subnet (4 total):" is the total count necessary?
- (Cosmetic - Consistency and standards) Network Map mode: This could be a QT thing, but radio buttons are generally used for "select 1 out of all" and square checkboxes for "select as many as you like". Radio check-buttons look funny.
- (Minor - Consistency and standards) Network Map mode: don't toggle "Show" and "Hide" as the user selects or deselects a checkbox. It's fine to use "Show" all the time. Users know that when it is unchecked, it means Don't Show.
- (Minor - Aesthetic and minimalist design/Consistency and standards) Network Map mode: the additional information about the selected subnet is useful, but it takes up precious screen real estate, and it's probably not used all the time.

How about popping up that info when the user mouses over a subnet title bar in the display?

- (Good - Help and documentation) Host Asset Values: The hint “Select host group to view asset values” is good. It tells me what that blank space is for.
- (Minor - Help and documentation) Host Asset Values: Similar hint would be nice for the panel below. I couldn’t figure out what that’s for until I started changing asset values.
- (Major - Error prevention) Host Asset Values: The lower panel should be non-editable. I accidentally edited it and confused the UI.
- (Major - User control and freedom) Host Asset Values: I can view the pending changes in the lower panel, and I can commit the changes by clicking “Apply asset...”, but the only way I can cancel is manually undo each entry in the upper panel.
- (Good - Recognition rather than recall) The +- buttons remind users the values are editable.
- (Major - Recognition rather than recall) Can I select a host (e.g., adam) and see which groups it belongs to by highlighting the display?
- (Minor - Aesthetic and minimalist design) Perhaps the View Hosts by Group and View All Hosts could be combined. If a Host Group is selected in the display, filter the list to include only the hosts in the host group. Otherwise, display all hosts.
- (Major - Help and documentation) Host Group Filters: skipped over because it’s not clear what this is for.
- (Minor - Recognition rather than recall) Network Information: it would be great to provide hyperlinks between hosts’ vulnerability list and detailed vulnerability information.
- (Minor - Recognition rather than recall/Flexibility and efficiency of use) Network Information: the host list could be coded (color, number) to indicate which host has vulnerability, how many, and what access level. That way, I don’t have to select every host to find the hosts with vulnerabilities or determine which hosts are more at risk than others.
- (Minor - Recognition rather than recall/Flexibility and efficiency of use) Network Information: switching between Hosts list and Vulnerabilities list causes the lists to auto-scroll back to the top. That means, if I was looking at a host and then switched to look up a vulnerability, I would lose my place when I came back to the Hosts list.

- (Minor - Help and documentation) Summary Plots: could not select multiple versions in the timeline to compare as suggested.
- (Good - ) The Attack Graph mode is really well done. The UI explains the how and why every step of the way. I really like this control and display.
- (Minor - ) Perhaps the “Add a new vulnerability” button belong to the Network Information:Vulnerabilities panel better?
- (Minor - Aesthetic and minimalist design) The Recommendation screen is too small and hard to read. I would suggest reducing this control panel to just “Attacker Depth” and “Recommendations”. Keep it simple. The Exploitable vulnerability information could probably be embedded in the Host Groups’ popup. It’s too much and too detailed, I think.

### User 3

#### Display Controls and Main Window

- Major: In general, I don’t care for the white squares in the center of each treemap square. I think that they clutter the display, especially with small groups. For all interactions (tool tips may be an exception) it would be easier to be able to click anywhere in the treemap square instead of having to be precise enough to hit the small dot. I did like that they turned colors to show the different levels in the attack graph, but you may be able to do that just as effectively by coloring the arrows themselves.
- Minor: The GUI should automatically ask to open a file upon launching, since this is the only thing a user can do at that point anyway.
- Minor: The title bar (or somewhere else) should show which network model is currently open.
- Critical: On the various TN networks I got an exception when trying to show the attacker step 3 or 4.
- Major: When selecting the attacker starting subnet, the subnets listed by name without any other information. This requires some specific knowledge about the data if the user wants to choose another network. It may be easier to bring up the display and allow the user to click on a subnet to select that as the starting location. The GUI should also allow a user to change the attacker starting location without going through the process of loading a file.
- Cosmetic: “Show All Within Subnet Reachability” difficult to parse.

- Major: The Subnet Reachability view is very difficult to understand because of all the arrows. One idea is to have a “show reachability” mode and only show reachability for a particular reachability group when that group is moused over. Another would be to show a single arrow for the entire subnet when all groups in the subnet have the same reachability to or from a particular target. You would somehow have to make it clear that this arrow applies to the entire subnet.
- Minor: “Auto-Size Subnets” view option text should be changed to something that indicates how the subnet sizes are being chosen (I assume it is number of hosts).
- Minor: Need a “grid” or “tile” subnet layout option
- Minor: Clicking on a subnet should select it. As is, you select reachability groups by clicking on them but subnets are selected only in the drop down list.
- Major: The GUI needs a “zoom to fit” option. Without it, autosizing subnets becomes mostly useless because they subnets then take up too much room to all fit onto the screen at once.
- Major: Either the “Reset Display” should reset the subnet sizes (after an auto-size) or there should be a separate option for uniform subnet sizing. Once subnets have been autosized, the user must resize them all by hand to undo the operation.
- Minor: Graying out of buttons makes them look like they are not selectable. There should be three distinct visual states: Not Selectable, Not Active, Active.
- Minor: Subnet Reachability panel on left was not expandable to fill the entire vertical space. This left only a small space for the (rather long) list of hosts.
- Minor: “+” button for host asset value is confusing. Either make the + and - actually do something or remove it and put a text field that looks editable in its place
- Critical: List of pending changes for asset value is editable (though it doesn't look like it is). Making edits here causes an exception.
- Minor: It would be nice if individual changes had a small “X” next them that you could click to reset the changed value.
- Minor: Some hosts are listed by IP, others by name. Especially when viewing all hosts, it can be difficult to see where they go (which subnet)
- Major: Selecting options from mouse-over menu has no effect
- Minor: There is no way to see subnet IP address range. This makes it difficult to get a feeling for where hosts belong.

- Critical: There is no way to see how subnets are related to each other or where filtering devices are. It would be nice to show layer 3 infrastructure devices (i.e. routers and firewalls) so that a user could see where firewall rule changes could be made.
- Major: Should DoS and Other levels of access be shown? They are ignored for the purposes of the attack graph and showing them breaks up the subnets into more reachability groups, causing more clutter when reachability or the attack graph is shown. They can be excluded, but then they aren't shown at all. It would be nice to fold them into the "No Access" group. Similarly, it would be nice to be able to fold the "User Access" blocks into the "Root Access" blocks.
- Minor: Network Model Timeline should be completely grayed out or not present at all when not in use.
- Major: Except for actual reachability traces, reachability groups that are all the same (i.e. their reachability differences do not affect the attack graph and they have the same type of vulnerability) should be put into the same group. This would lead to less clutter. For actual reachability traces, you may need to show the different groups, but that could be done dynamically.
- Minor: Showing reachability would be easier if you had a "mouse-over" mode so you could turn it on and then move the mouse around to see the inbound or outbound reachability for the particular block you were over.
- Minor: Reachability would be more useful if I could find out what ports were being reached.
- Good: Transparent windows when moving is nice.

#### Network Information and Filters

- Minor: View Vulnerabilities and Trace Path from Attacker links do not appear to do anything (I think they jump to the relevant sections if those sections are not currently visible). This was confusing to me. Could they be put on separate tabs?
- Minor: I could view the entire network or individual reachability groups, but could not view individual subnets. As mentioned above, it would be nice to be able to click anywhere in a host group instead of having to click on the white square. Right now, clicking on a host group (but not on the white square) resets the display to the entire network, which is somewhat counter-intuitive.
- Major: Clicking on divider between the host list and the information on that host made it jump down, but I couldn't adjust it any further. This looks like a bug.

- Minor: Host Group Filters should just have the filters, not a single checkbox you need to select to first enable them. It should then have an “Enable All” button and a “Disable All” button.
- Minor: Vulnerability list entries should be color-coded to match the colors in the treemaps.
- Minor: I should be able to filter the vulnerabilities that are in the list so I only see those of a certain type.

### Attack Graph

- Minor: I should be able to click on a network model snapshot and revert back to that one instead of having to use the arrows.
- Major: It would be nice if there was a way to see where the various recommendations were in the treemaps. Numbers in the squares containing the vulnerabilities being patched would be one way. Otherwise, it is not clear what a particular recommendation is doing.
- Major: The Network Model Timeline looks like a slider where the icons should be moveable with the mouse. One way to do it would be to have a row of boxes where unused boxes are grayed out and boxes are used from left to right as new models are created.
- Minor: It would be nice to be able to mouse over a recommendation and see the effects it would have. Otherwise, I have to select it, apply it, and then undo it in the Network Model Timeline to look at another one.
- Major: It would be easier to choose a zero-day vulnerability if I had a good way of finding out how many of what ports were open on the network and in which subnets or reachability groups.
- Minor: When arrows are drawn for attack graph, if they are drawn to a box obscured by another box, the arrows are drawn in front of everything. This makes them appear to point to the overlaying box.
- Good: I like the use of color to coordinate the attack graph steps in the left panel with the treemap view.

### Summary Plots

- For the anon network in “Assets Captured vs. # of Hops”, there’s a 3rd hop shown but the number of assets captured doesn’t change.
- Critical: Clicking didn’t work for me to select multiple network models. The only way I could do it was to Ctrl-Right Click and this only worked for when I selected the models in a certain order.

- Major: Clicking anywhere in the Network Model Timeline automatically selected the last model and deselected any others.
- Major: The timeline labels the recommendations based on their number at the time they were selected. I ended up with “Recommendation 1” followed by “Recommendation 3” followed by “Recommendation 1”. The recommendations should probably be given numbers that do not change so that recommendation 4 always refers to the same recommendation.

#### User 4

- Major: No documentation on various buttons/features/etc. This could be remedied with a manual and/or with plentiful tooltips and a searchable help function of some sort. (Don’t feel that you need to address this; this is somewhat obvious and known and can’t really be tackled until the interface is finalized to some extent.)
- Major: User is forced to choose attacker starting location during network load. (User control / Freedom) User must reload network model from disk to change attacker starting location. (User control / Freedom) Progress bar during network load doesn’t reveal that attack graph is being built at that time. (Visibility) Consider decoupling network load from graph generation. (Me adding an ability to get host to reachgroup mappings would help.) This would mean all nodes would show up as uncompromisable in the network view, or they’d show up as compromisable based on their vulnerabilities, without regard to a particular attacker location. If it does remain this way, change the text on the progress bar as the work proceeds from loading the network model to generating the attack graph to siphoning it through the SWIG straw, so the user knows what’s really taking so long.
- Visible subnets don’t include all subnets; RedSeal documentation lists 28, but only nine are shown. (Visibility) What happened to the others? Are they omitted because they have only multihomed hosts on them?
- Minor: Multihomed hosts are shown multiple times; once for each subnet they’re contained in. (Clarity) Consider showing multihomed hosts as standalone “floating” entities, connected to the (multiple) subnets to which they’re linked.
- Minor: Subnets are placed in a 1-D column or 1-D row. (Visibility) Consider a default layout that is a grid.
- Moderate: Connections/links between subnets are not shown. (Visibility) Show those links, and perhaps use them to dictate the default initial layout of the subnets.



- Moderate: Subnet size is not by default proportional to the total asset value in the subnet. (Visibility) Subnet size option is available, but is in a non-obvious location. I call it “difficult to find” simply because 99% of the tool’s functionality is exposed by buttons in the main window. I became conditioned to expect all functionality except for the “Load” command to be present in the main window, and didn’t even think to check the menubar for further options. I agree that the “View” menu is a good place for these options, but I don’t know what to do about the user’s drifting attention. Consider making size-by-weight the default option.
- Major: The “Reset Display” option didn’t do anything for me. (User control) Make it actually reset the display in some way.
- Major: By resizing the top corner of a subnet, one can resize the top above the top of the viewport. It is then impossible to select/move/resize that subnet. (Error prevention; User control) (Note: This happened on Mac; I can’t replicate this on PC.) Forbid the user from resizing above the top of the viewport, or have the viewport dynamically resize with the user on all four edges, not just the right and bottom edges.
- Cosmetic: Context menus for groups appear only when hovering over the “dot”; the “dot” doesn’t mean anything else (Match real world) Consider removing the dot entirely and having the hover menu appear when over any part of the group. Consider adding individual (all-black, less prominent) dots for hosts, to illustrate visually the number of dots in the group when the group is hovered over.
- Minor: A subnet can only be selected by using the drop-down list on the left pane. (Flexibility; Consistency) I expected to be able to click on the subnet itself to select it on the left pane. Consider removing the drop-down list entirely.
- Minor: Many “select one of many modes” options, such as “Network Model / Attack Graph / Summary Plots” are shown as buttons instead of tabs. (Consistency) The “Network Info” panel is selectable while in Summary Plots, but it doesn’t do anything there. (Control) Make them tabs at the top of the screen, so it’s clear the user is choosing between one of three modes. Similarly, perhaps make “Display Controls” and “Network Map / ...” two subtabs that appear only for Network Model and Attack Graph.
- Minor: If two subnets overlap and “show all (in/out) subnet reachability” is chosen such that arrows are drawn to/from the occluded subnet, the arrows appear at the top level and look like they’re going to/from the subnet that’s on top. (Error prevention) (I don’t have any good ideas on how to fix this one, I’m afraid. You could forbid subnet overlap, but that’d be tricky, because you want users to be able to drag subnets around.)

- Very Minor: Users must use the scroll bars to move around the viewport horizontally. (Control) Consider allowing the user to drag through the view by dragging a non-subnet area.
- Minor: The “Network Model Timeline” is inconsistent. In Network Map and Attack Graph views, users must use the arrows and cannot click on individual elements of the timeline; in the Summary Plots view, users cannot use the arrows and must click on individual elements of the timeline. (Consistency) Eschew the arrows and just let the user click on the chosen entry in the timeline in all views.
- Major: In Network Map and Attack Graph views, you can’t tell which element in the timeline is active. (Consistency, Error Prevention) In particular, when two items are on the timeline and the second one is selected, the “Root Model” label does not get grayed out, so both entries on the timeline look equally likely. For consistency, consider using the same green box that’s used for the Summary Plots.
- Minor: Vulnerability descriptions don’t appear to be visible anywhere. Find somewhere that it makes sense to show the descriptions of the vulnerabilities, and show ’em there.
- Insect: In the RedSeal network, adding a remote-to-root on 22/tcp shows several hosts in the 10.1.2.0/24 subnet as exploitable, but no path from the attacker to those nodes is shown. (Bug?)
- Major: Making a change from the root node of the network timeline destroys all later changes. (Error prevention) Make it more obvious to the user that this will happen; allow the user to cancel.
- Insect: Attacker reachability trace doesn’t work at all; nothing happens at all when I try it. (Bug?) Is there something wrong with my setup? Does it work for you? I tried on both Mac and Windows.
- Minor: “Add a zero-day” is not part of the Network Map section. (Consistency) Consider unifying all of the stuff that alters the network model in the “Network Map” area, or some related tab. (You already have asset value changes over there, and that interface is pretty slick.)
- Minor: User must self-identify ports for the reachability trace. (Efficiency) Consider a dropdown list of open ports on the host for the user to choose from.
- Insect: Loading redseal, then anon, then redseal seemed to create a bizarre blank dialog. (Bug?) (I didn’t try to replicate it.) Shortly thereafter it segfaulted. I’ll have to track that down.
- Insect: Loading TN1-A and then going to the attack graph view, I see that there are four steps. The first two show up; clicking on the third causes an exception. Does this happen for you too?

## User 5

- Good: Using a treemap is a good compact way to display many hosts and subnets.
- Good: Similar hosts as far as security are concerned are grouped in one region.
- Good: The GUI is extremely quick to draw links and hosts and when moving subnets or showing incoming/outgoing links.
- Good: The layout is clean and simple for attack graph and reachability.
- Good: Following recommendations and adding zero-day vulnerabilities leads to more networks and the result is cumulative. It is easy to work with up to five or more different networks.
- Cosmetic: The colors used to show compromise level aren't very unique. You might use colors going from green to blue to red to designate the compromise levels going from none to other to DoS to user to root. Green could be none, DoS could be black, and user and root could be orange and red of different shades.
- Major: It is difficult to know what subnet you select for attacker starting location. Before you see the network and its subnets you have to select the attacker starting location. For example on the RedSeal network there are many subnets and you need to remember which one interfaces to the external world. It would be better either to select external which defaults to the connection to the internet, or to select the starting location after the network map is displayed. It might be possible to show external connections to the internet on the network map and select this as an external attacker source. The user needs some clues when selecting the attacker starting location.
- Cosmetic: The label for the attacker starting location selection window is cryptic. Change the label to "Specify Attacker Starting Location".
- Major: It is difficult to find information using the current upper three tabs (Network Map, Attack Graph, Summary Plots) combined with the bottom three tabs (Display Controls, Network Information and Filtering). Information is scattered in different places and accessed in different manners. Sometimes you click on a node in the display and sometimes there is a pull-down menu. It is difficult to predict what combination of upper and lower tabs produce, the names for the bottom tabs are non-intuitive, and the bottom tabs really are independent of the top three tabs and provide extra information. One solution is to eliminate the bottom two tabs, combine the current information together to fit into the Network Map and Attack Graph Displays, and put the filtering selection into a "Prefs" or "Options" pull down menu at the top of the display

next to the current “File” and “View” Menus. The following eight changes would make this happen:

- Major: Move the filters settings from a tab to a pull down preferences menu at the top of the display as shown in Figure 1.
  - Major: In the Network Map display clicking on a subnet should select it and you should get rid of the pull down menu as shown in Figure 1.
  - Major: Get rid of the bottom two selector tabs and move the host group information from the old Network Information Display to the bottom of the Network Map display as shown in Figure 1.
  - Major: In the list of vulnerabilities in the new Network Map Display separate those used by the attacker from those that aren’t used as shown in Figure 1.
  - Major: In the list of vulnerabilities in the new Network Map Display when you click on a vulnerability, it should set the port for the trace function to the port of that vulnerability.
  - Major: Remove the exploitable vulnerabilities pane from the attack graph display as shown in Figure 2.
  - Major: Remove the host asset values pane from the Network Map Display and put it into the Attack Graph display as shown in Figure 2.
  - Major: Move the zero-day selector down in the Attack Graph pane as shown in Figure 2.
- Minor: Layout files are not shown when you load a network model. When you load a network model you should have the option of using an available layout file, if one exists.
  - Major Bug: If you drag any subnet to the top of the display so its name is no longer visible, you can no longer move this subnet. Fix this bug, so the name cant move outside the display area.
  - Cosmetic: In the network map mode, visually indicate the subnet that is selected.
  - Major: Show outgoing links from the attacker node in the Network Map mode. This is important because determining hosts that are exposed to the attacker is essential, and these can not currently be found.
  - Major: Host Groups compromised at DoS and Other Access levels duplicate Root and User access hosts. The display is misleading because hosts compromised at both the root level and other level are displayed twice and make the subnets look larger than they really are. Hosts should only be put into one group representing the highest level of compromise. A host should be put in the “Other” group only if it is not in the “Root”, “User” or “DoS” group. A

host should be in the “DoS” group only if it is not in the “Root” or “User” group.

- Minor: Don’t show “Other” access group by default. If the previous problem is fixed, this is no longer a problem. Because the “Other” group complicates the current display, I currently set the current display to exclude these groups.
- Minor: Host Group Filter settings keep getting reset. These settings keep getting reset to the default settings whenever I do anything and this is extremely annoying. They should stay put because they apply to all networks and are similar to display preferences. They should not be reset when you apply a recommendation, add a zero-day vulnerability, or load another network model.
- Network Description: There is no incoming connectivity to enclaveINT for the anon network. Fix the network description unless this is an air-gapped network.
- Minor: I have a hard time with the “Show all Within Subnet Reachability” check boxes. This may be me, but these check boxes in the Network Map/Display Controls mode are confusing because they show what will happen if you click on the checkbox but the checkbox seems to be saying this is what you currently see. These are conflicting indications. A yes/no toggle might be more intuitive.
- Major Bug: Attacker Path Trace not shown for host 192.168.1.2 in anon network. In the anon network, if you click on the hosts compromised as root in subnet “enclaveDMZ” which is host 192.168.1.2, in the Attack Graph/Network Information and Filters Mode, you can see the ports that are open. If you type in port 22 (the port with the vulnerability used) into the port number box for the attack trace in the “Network Information and Filters” display, you get “Host not reachable on port 22”. This is clearly incorrect because in the Attack Graph/Display Controls mode, if you click on this host it shows three vulnerabilities on port 22.
- Minor: Can’t distinguish between recommendations and exploitable vulnerability scrolling pane contents. This is not a problem if you follow the above recommendation that change the Attack Graph and Network Map displays but it is a problem with the current display. In the Attack Graph/Display Controls mode I get confused about the pane that relates to the recommendations and the region that relates to Exploitable vulnerabilities. When you click on a host, the information below the recommendations region changes to show the vulnerabilities on a host but this looks suspiciously like the information in the recommendations panel. I sometimes look in the wrong place to see the vulnerabilities on a host and get confused. This is most confusing if you scroll the Recommendations display region and the Exploitable Vulnerabilities regions since they show exactly the same “List by Host” and “List by Vulnerabilities” selectors. You can move these apart more and use more of the real estate on the left part of the window, change the background color for these two regions, or put a darker rectangle around the two regions.

- Future Work: Patch exploitable vulnerabilities. It would be great to add a check box that would let a user patch host groups and see the results. This would be similar to the effect of selecting a recommendation, but you could determine the recommendation yourself. An easier alternative is to add to the recommendations some to patch each host group that is compromised at the root level.
- Major: Need a stronger visual indicator of location on the Network Timeline. I never know where I am on the network timeline. You explained that it is the last icon that is colored, but that is not intuitive. You might draw a red or black circle around the icon representing the network that is currently selected.
- Major: Selection of the Network Model in the timeline is not intuitive. The network is selected using the upper and lower arrows, but I keep trying to click on the icons for the networks. I would allow a user to click on the icons. This is a major problem because clicking on the icons works when displaying summary plots but not in the other modes. This needs to be consistent across modes.
- Future Work: No guidance in adding a new zero-day vulnerability. When the user adds a new zero-day vulnerability, it is difficult to know which one to add or how much damage it will do. I would add an option like “Worst-case zero day vulnerability” and this would find the port that does the most damage. This needs to be supported by Kyle. You should keep the current functionality and allow a user to select a port.
- Major: Allow “All Ports” option when selecting a new zero-day vulnerability. The worst possible attacker model is someone who has a vulnerability for every server running on the network. Ask the user to select this attacker model or to enter a wildcard or range in the port selection for the new zero-day vulnerability.
- Minor: Sizes of subnet boxes could reflect number of hosts in the subnet. It would be good if the sizes of the subnets initially reflected the number of hosts in the subnet or their asset value sum.
- Major: Switching between networks on the network timeline doesn’t work. In the Summary Plots/Assets Captured vs # Hops mode the arrows at the bottom of the display that move along the timeline are grayed out and don’t work. It sometimes works to change networks by clicking on the network icons. This is with 5 networks created by applying three recommendations and adding one vulnerability.
- Cosmetic: Name versions of networks created. Instead of naming the different networks with “Recommendation 1 applied” or “Vulnerability Added” you might allow a user to add a short descriptive name.
- Minor: When you filter out reachability groups that are “other”, they still show up in the pie chart.

- Major: The Assets captured vs. number of unique exploits used chart is wrong. This should be made to work by drawing maybe 100 or so greedy examples of the attacker that prefers to go through firewalls first and showing all the plots on top of each other. The plot should show the number of unique exploits required not the total number of vulnerabilities or exploit steps. Alternatively this chart could be dropped, but it is interesting.
- Major: The assets captured vs attacker effort expended plot isn't correct. Talk to Kyle about fixing this. You could set the "low" effort to a delta of 1.0 and the high effort to a delta of 3.0. The plot should thus step up at 1 (low effort attacks), 3 (high effort and low effort attacks), 4 (combinations of one low and one high effort attacks), and 6 (two high level attacks) if there are two steps.
- Cosmetic: Increase font size for summary plots.
- Major: Recommendation is inconsistent with Exploitable Vulnerabilities List. In the anon network, the first recommendation is to patch four vulnerabilities on main-server.example.gov but the exploitable vulnerabilities list right below the recommendations shows only one vulnerability on this host. This is inconsistent and should be fixed.
- Cosmetic: The check box next to "Apply selected recommendation" never changes. Shouldn't this be a push button?
- Minor: Host names are missing for the redseal network. If host names are missing, fill them in with a unique identifier (often IP address is OK).
- Future Work: Bad recommendations for redseal network. The recommendations for the redseal network do not include the obvious choice to patch the web servers at the bottom of subnet 10.5.1.0/24 and the single vulnerability on port 22 for the node at the bottom of subnet 172.16.3.0/24. These should be available or you should be able to click on a host and patch it.
- Major: Percentages are wrong for hosts compromised for Attack Graph Display. For the redseal network the percentages are 500.0% (10/2) for 1 step, 1450.0% (29/2) for two steps and equally wild for other steps.
- Major: System crashed when I loaded TN1 after exploring anon and redseal networks. It displayed the network incorrectly with a white dot in the upper corner of the single rectangle for each subnet and another white dot at the center. I could replicate this incorrect display by opening the redseal network, applying the first recommendation, adding a zero-day vuln on port 22, and then loading TN1, but I couldn't make the system crash.
- Cosmetic: Put the name of the current open file at the top of the display.
- Network Description: Firewall interfaces are considered hosts in TN1-\*. On TN1-A and other TN1 networks there are extra hosts in subnets for firewall

interfaces. For example in subnet B (DMZ) there is a host labeled “FW1 DMZ intf” and “FW2 DMZ intf”. These shouldn’t be on the subnets unless they can be accessed from a control port and have an actual IP address. This is probably a problem with setting up these test networks.

- Cosmetic: Move components of treemap in each subnet around to minimize length of between-subnet links.
- Minor: It is difficult to determine hosts in graph that will be patched by each recommendation. It is difficult to determine which hosts are to be patched for the different recommendations, especially to see where the hosts are on the display. I have to go to a different mode (network information and filters) and randomly mouse on different nodes to find those host numbers that are in a patch rule. It might be better to have some way of highlighting the hosts that will be patched with different recommendations or to select the hosts using the list in the recommendations pane.
- Network Description: In network TN1-E, the asset value for host 5 should be 100, not 10.
- Cosmetic: When you click on the windows, shrink button it shrinks down to almost nothing instead of an intermediate size.
- Minor: Attacker starting location should default to outside.
- Minor: What is the upper “View” menu there for? It probably should be a toggle that takes away and then restores the links, but I’m not sure what it is there for.



# Appendix C

## User Evaluation Summary

### Critical Items

1. It is difficult to find information using the bottom selector buttons (Display Controls, Network Information and Filters) combined with the upper three mode buttons.
2. Interaction with the network model timeline is not intuitive.
3. It is difficult to determine which subnet to select for the attacker starting location.

### Major Items

4. For the Network Map mode, subnets are not directly selectable.
5. Subnets can be moved or resized past the top of the display area and become immovable.
6. The subnets defined in the network model are not all visible.
7. There is no “All Ports” option provided when selecting a new zero-day vulnerability.
8. Host Groups compromised at DoS and Other Access levels duplicate hosts compromised at Root and User levels.

9. The summary metric plots for the attack graph are not entirely accurate.
10. The recommendations are inconsistent with the information in the Exploitable Vulnerabilities pane.
11. Subnets do not all fit within display area when auto-sized.
12. There is no way to undo auto-sizing.
13. The lower host asset value pane is (and should not be) editable.
14. No documentation on the various buttons, features, etc is included.
15. Firewall interfaces are shown as hosts within subnets.
16. Connectivity between subnets is not immediately apparent.
17. There is no option for saving modified network models.
18. Subnet reachability is difficult to view and understand.
19. The attacker path trace is not available for all hosts.
20. Reachability groups with different compromise levels cause unnecessary clutter.
21. There is no way to view open ports per host or for the entire network.

### **Minor Items**

22. Links that are drawn between overlapping subnets are visible even though they may be drawn to/from an occluded node.
23. Subnet size is not by default proportional to the total asset value in the subnet.
24. There is no legend explaining the colors used for compromise level.
25. The white host group icons are not particularly useful.
26. Subnets can be positioned into a single column or row, but there is no grid layout option.

27. The Network Information pane can only show information for all hosts or a single host group.
28. Switching between host and vulnerability lists in the Network Information view is tedious.
29. When the application is first started, the only possible action is to load a network model.
30. Layout files are not shown when you load a network model.
31. The network model icons on the timeline are not usefully labeled.
32. The filter settings are reset whenever a new model is generated.
33. The effect of the “Reset Display” menu option isn’t obvious.
34. The control panes in side panel are not resizable.
35. Grayed out buttons can be confusing since they usually indicate a feature is disabled.
36. The IP address ranges for subnets are not shown.
37. The summary plots are inconsistent with the filter settings.
38. The host filters panel should have better controls.
39. The host and vulnerabilities lists should be color-coded and filterable.
40. There is no support for previewing the effect of a recommendation before applying it.
41. The buttons for editing host asset values are confusing.
42. Users must use the scroll bars to move around the viewport horizontally.
43. The menu options should have keyboard shortcuts.
44. The Load Model dialog should remember the previous directory.

## Cosmetic Items

45. The check boxes in the Subnet Reachability pane are inconsistent and confusing.
46. The colors used to show compromise level aren't very unique.
47. Timeline is not necessary when there is only a single network model version.
48. The extra information in the Subnet Reachability pane is not necessary.
49. The label for the attacker starting location input dialog is cryptic.
50. The labels for the summary plots can be hard to read.
51. The check box next to "Apply selected recommendation" never changes.
52. There is no indication of the currently open file.
53. Submenus are hard to navigate and should be avoided.
54. The button tooltips for the network model open dialog are unnecessary.
55. The title bar contains unnecessary information.

# References

- [1] B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: making effective use of 2d space to display hierarchies. *ACM Transactions on Graphics*, 21(4):833–854, 2002.
- [2] CVE. Common Vulnerabilities and Exposures Dictionary. <http://cve.mitre.org>. Accessed 20-May-2008.
- [3] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *Proceedings Computer Security Application Conference*, pages 121–130, 2006.
- [4] S. Jajodia and S. Noel. *Topological vulnerability analysis: a powerful new approach for network attack prevention, detection, and response*. Indian Statistical Institute Monograph Series. World Scientific Press, 2007.
- [5] JFreeChart. <http://www.jfree.org/jfreechart>. Accessed 20-May-2008.
- [6] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings 2nd International IEEE Visualization Conference*, pages 284–291, 1991.
- [7] JUNG. Java Universal Network/Graph Framework. <http://jung.sourceforge.net>. Accessed 20-May-2008.
- [8] R. Lippmann and K. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, Lexington, MA, 2005.
- [9] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham. Evaluating and strengthening enterprise network security using attack graphs. Technical report, MIT Lincoln Laboratory, Lexington, MA, 2005.
- [10] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, and R. Cunningham. Validating and restoring defense in depth using attack graphs. In *Proceedings MILCOM*, Washington, DC, 2006.
- [11] R. Lippmann, L. Kukolich, and E. Singer. Lnknet: neural network, machine-learning, and statistical software for pattern classification. *MIT Lincoln Laboratory Journal*, 6(2):249–268, 1993.

- [12] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to common vulnerability scoring system, version 2.0. <http://www.first.org/cvss/cvss-guide.html>.
- [13] C. Mumford-Valenzuela, J. Vick, and P. Wang. *Heuristics for large strip packing problems with guillotine patterns: an empirical study*, pages 501–522. Metaheuristics: computer decision-making. Kluwer Academic Publishers, Norwell, MA, 2004.
- [14] T. Munzner. 15 views of a node-link graph: an infovis portfolio. <http://www.cs.ubc.ca/~tmm/talks.html#google06>. Accessed 20-May-2008.
- [15] Nessus. Tenable Network Security. <http://www.nessus.org/nessus>. Accessed 20-May-2008.
- [16] J. Nielsen. *Heuristic Evaluation*. Usability Inspection Methods. John Wiley and Sons, New York, NY, 1994.
- [17] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings ACM CHI 1990 Conference*, pages 249–256, Seattle, WA, 1990.
- [18] S. Noel and S. Jajodia. Understanding complex network attack graphs through clustered adjacency matrices. In *Proceedings Computer Security Applications Conference*, pages 160–169, Tucson, AZ, 2005.
- [19] NVD. National Vulnerability Database. <http://nvd.nist.gov>. Accessed 20-May-2008.
- [20] D. Phan, L. Xiao, R.B. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proceedings IEEE Symposium on Information Visualization*, pages 219–224, 2005.
- [21] Piccolo Toolkit. <http://www.cs.umd.edu/hcil/jazz>. Accessed 20-May-2008.
- [22] RedSeal Systems Inc. <http://www.redseal.net>. Accessed 20-May-2008.
- [23] B. Shneiderman. Tree visualization with tree-maps: a 2-dimensional space filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.
- [24] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006.
- [25] Skybox Security Inc. <http://www.skyboxsecurity.com>. Accessed 20-May-2008.
- [26] Symantec Corp. Internet Security Threat Report. <http://www.symantec.com/business/theme.jsp?themeid=threatreport>. Accessed 20-May-2008.