Visual Correlation of Network Traffic and Host Processes for Computer Security

Glenn Allen Fink

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Dr. Chris L. North, Chairman
Dr. James D. Arthur
Mr. Randy C. Marchany
Dr. Scott F. Midkiff
Dr. Deborah G. Tatar

15 August 2006
Blacksburg, Virginia

Keywords: Computer Security, Information Visualization, Human-Computer Interaction, Software Architecture, Software Design, Insight-Based Method

Visual Correlation of Network Traffic and Host Processes for Computer Security

Glenn A. Fink

Abstract

Much computer communications activity is invisible to the user, happening without explicit permission. When system administrators investigate network communications activities, they have difficulty tracing them back to the processes that cause them. The strictly layered TCP/IP networking model that underlies all widely used, general-purpose operating systems makes it impossible to trace a packet seen on the network back to the processes that are responsible for generating and receiving it. The TCP/IP model separates the concerns of network routing and process ownership so that the layers cannot share the information needed to correlate packets to processes. But knowing what processes are responsible for communications activities can be a great help in determining whether that activity is benign or malicious.

My solution combines a visualization tool, a kernel-level correlation engine, and middleware that ties the two together. My research enables security personnel to visually correlate packets to the processes they belong to helping users determine whether communications are benign or malicious. I present my discoveries about the system administrator community and relate how I created a new correlation technology. I conducted a series of initial interviews with system administrators to clarify the problem, researched available solutions in the literature, identified what was missing, and worked with users to build it. The users were my co-designers as I built a series of prototypes of increasing fidelity and conducted usability evaluations on them. I hope that my work will demonstrate how well the participatory design[10] approach works.

My work has implications for the kernel structure of all operating system kernels with a TCP/IP protocol stack and network model. In light of my research, I hope security personnel will more clearly see sets of communicating processes on a network as basic computational units rather than the individual host computers. If kernel designers incorporate my findings into their work, it will enable much better security monitoring than is possible today making the Internet safer for all.

This work is dedicated to my Lord and Savior Jesus Christ without Whom none of it would be possible or meaningful. To properly tell you what He means to me would take more than the space of this dissertation and that treatise would have far more meaning. Reader, I pray that He would find you and that you would come to know Him. – Glenn Fink

ACKNOWLEDGEMENTS

First of all I thank my wife for her outstanding editorial skills without which I would be continually using "which" instead of "that" in restrictive clauses and drowning in excessive commas. No thanks could be enough for her part in my life and work. My thanks to my good friend and "personal statistician," Peter Parker, for his help and encouragement in designing the usability study of HoNe. Pete, I could never repay the hours you spent patiently chatting with me about experimental design at your own expense. You are a blessing. I would also like to thank Virginia Tech's security officer and the system administrators I interviewed for their patience, enthusiasm, and guidance as I shaped this product to fit their needs.

Special thanks is due to Ricardo Correa, Bob Ball, Nipun Jawalkar, and Paul Meussig for their help interviewing subjects, coding prototypes, and conducting usability evaluations. Thanks goes to my advisor, Dr. Chris North, for his undying enthusiasm. From now on I'll call you "Chris" like you always wanted. This research was supported in part by a National Science Foundation Integrated Graduate Education and Research Training (IGERT) grant (award DGE-9987586).

O the depth of the riches both of the wisdom and knowledge of God!
How unsearchable are His judgments, and His ways past finding out!

For who hath known the mind of the LORD?
Or who hath been His counselor?

Or who hath first given to Him, and it shall be recompensed unto him again?

For of Him, and through Him, and to Him, are all things:
To Him be glory for ever. Amen.

Romans 11:33-36 (KJV)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Imagine a user starts up her computer and sees an alert window pop up first thing saying, "New system updates are available. Download?" How does the computer know this? It checks on the network (usually via the World-Wide Web) and compares what is available to what is installed. However, she is only aware of this activity because the program informed her. Some other questions might be:

- What other communications activities are happening at any given moment that the user is unaware of?
- Who initiates these activities and for what purpose?
- Are any potentially malicious communications activities happening that are also invisible to the user?

Today's computers are designed to work as part of a network of computers. Perhaps it is more accurate to say that computers are now designed to be holders of communicating processes that work across a global network. Now, more than ever before, Sun Microsystems' trademark, "The network is the computer™," first articulated by John Gage[1], is true. More than ever before, the network defines the way we use our computers.

As a result, the person sitting in front of the machine may no longer be considered its primary user. The birth of client-server programs and the subsequent rise of peer-to-peer networks means that numerous persons may be invisibly using a computer even when it appears to be idle. Determining whether use of a computer is normal and authorized or malicious is a more important and difficult problem now than ever.

The way a person uses a computer is via one or more processes that do his work for him. These processes may communicate with other processes, both local and remote as needed, to carry out the user's instructions. Processes communicate over a network by sending information in short bursts called packets. Anyone on a network can see these packets, but determining whether they represent benign or malicious activity is something of an art form. Indications may come from packet contents (including the type of the packet and what port numbers it is using to access the computers where its processes live).

The problem is that the strict layered design of the TCP/IP networking model separates the concerns of network routing and process ownership. Thus, the layers cannot share information needed to trace a packet seen on the network back to the processes that are responsible for generating and receiving it. This makes true packet-process correlation impossible in all major operating systems without modifying the kernel. Concisely put, it is impossible to know at one place and time both the host address where a packet originated and what local process(es) will use the packet.

Knowing what process a packet came from and where it is going would provide a great deal of information that would be useful in uncovering malicious activity. Given today's software, a packet destined for port 80 on some computer is generally considered to be a web request. But nothing prevents some other program from running on port 80 rather than a web server. If a user can determine that a packet is destined for /usr/bin/httpd then she is relatively reassured that the packet is what it says it is. However, if the user finds that this packet is bound for /tmp/.../hackme.exe, she might have serious cause for concern.

This research is about helping users (particularly system administrators and security officers) to be able to see what processes each packet belongs to so that they can judge whether the communication is

benign or malicious. In the following discussion, I will talk more about this problem and what I have done to alleviate it.

1.1 Defining the Problem

When I interviewed system administrators (discussed in Chapter 2) I found they had no tool that directly supported fusion of host and network data. I wondered why no one had written a tool to correlate packets to processes, so I scoured the literature and commercial offerings (discussed in Chapter 3) for any tools that provided this integrated view of host and network activity. As I studied tool after tool in my literature survey, I found nothing that supported analysis across the host/network divide. This surprised me because the interviews convinced me that such a tool would be very useful, particularly to security monitoring.

The Host/Network Divide

Further investigation into the Linux kernel revealed the potential problem: The host/network divide is written into the kernel itself[2]. This is not only true of Linux, but of every other operating system that incorporates sockets according to the TCP/IP model, including Windows (NT and later), Mac OS X, all Unix-like operating systems, and a host of others including many embedded hardware systems. Because of Linux's open-source nature, I have confined my research to Linux implementations, but similar work could be done in any of the affected systems. Figure 1 shows a block diagram of Linux's networking stack. More detailed information on the Linux kernel is available in [3].

The networking stack in the kernel is organized into layers according to the TCP/IP model ([5],[6]) with application layer sockets at the top. Below the socket interface is the transport layer. This layer contains the TCP and UDP protocol modules that are responsible for mapping ports to processes and vice versa. Below the transport layer lies the network layer that makes protocol and routing decisions. The lowest layer above the hardware itself is the datalink layer that determines which interface packets belong to. Of these, the transport layer is the most important for mapping processes to packets. Visibility into this layer is what is lacking to allow development of tools capable of automatic correlation of packets to processes and vice-versa.

When an incoming packet arrives at the network interface, it finds its way up the protocol stack to the network layer where the machine makes a decision about whether the packet is addressed to it or not. At this layer, the source and destination IP addresses are known, but what process on the machine (if any) the packet belongs to is unknown.

If the packet is intended for a process on this machine, the transport layer's job is to deliver the information in that packet to the correct process. The transport layer does this by executing the callback function(s) stored in the socket data structure. Then the kernel wakes up the process, calls its callback function, and feeds it the data via an application-layer socket interface that looks to the process like an open file. Since processes may hand off sockets to each other or share them, the exact destination of the packet is determined dynamically. Partially because of this complexity, no programming interface exists that accurately exposes the identity of the receiving process. Although the Linux /proc filesystem keeps some of this information, it is not always accurate, especially with connectionless protocols like UDP.

In the network layer we can know the source and destination machine addresses but not what process the packet belongs to or whether it belongs to a process at all. Meanwhile, in the transport layer we can associate the packet to a running process or know that it belongs to none, but the source and destination host addresses are hidden. So the problem is that given all modern operating system kernels, we cannot at the same time and in the same place know where an incoming packet is coming

from and to what process it is bound. This lack of correlation I call the host/network divide, and it makes analyzing network communications much harder.

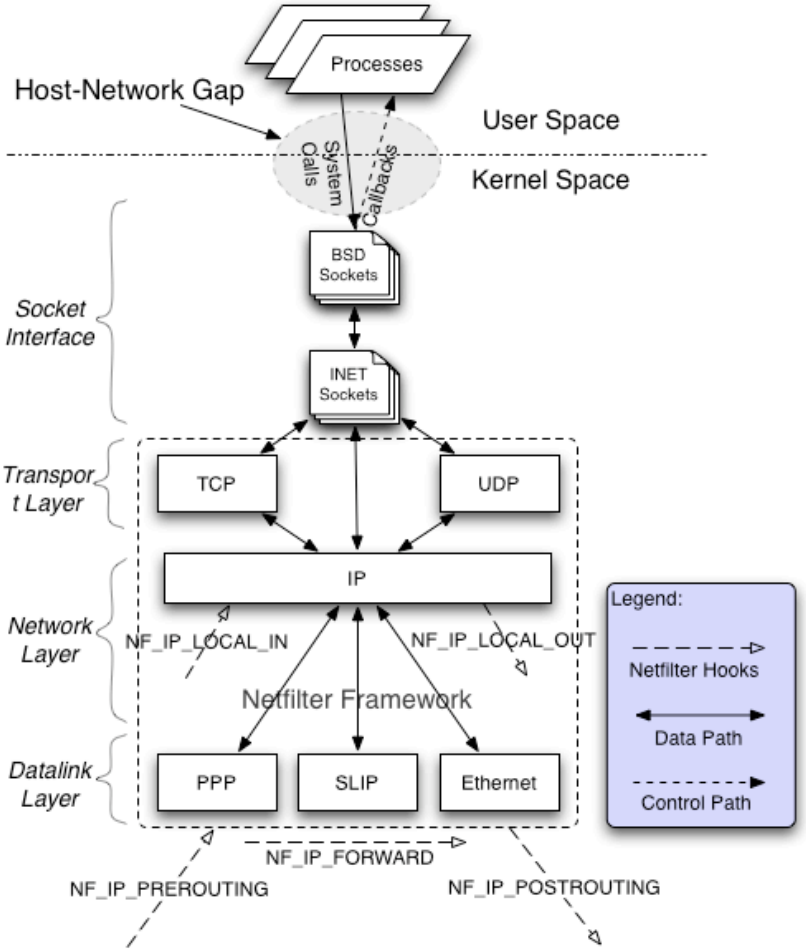


Figure 1: The Linux IP stack. Depicted are the various protocol stack layers, the extent of the Netfilter[4] framework, the five Netfilter programming hooks, and the location of the Host-Network Divide.

Human Dimensions of the Divide

The host/network divide is a fundamental separation between network and host that underlies most people's ways of thinking about networked computers. This separation has caused security practitioners and software designers to focus on either the computer or the network without considering the relationship of the events that pass between them. When investigating communication anomalies, system administrators must mentally fuse network and host information to develop a detailed model of the situation, but existing tools do not offer integrated support for this effort.

For example, supposing an administrator begins to see many incoming connections to TCP port 25 (the port for Simple Mail Transfer Protocol (SMTP)) on a subnet where he maintains no mail servers. First, he may try to determine whether the anomaly indicates an attack, a misconfiguration, or a security breach. Part of doing this is to isolate which process on the machine has this port opened. Usually this is done via netstat or lsof on Unix-like operating systems. He might also check the firewall and service logs and look at the file system. If he finds some indication that his web server has

suddenly sprouted an unauthorized mail service, he will know there was a compromise. But he will have to come to this conclusion by mentally piecing together data from the network with data from the host.

The way modern operating systems are organized forces people to look at either the host data or the network data separately—not both simultaneously, and not a correlated view. What is more, most people think of host and network as separate data sources regardless of the fact that some process running on a host somewhere must generate every packet seen on the network. Thus people tend to create tools and methods that compartmentalize hosts and networks rather than thinking of them holistically. Designers created this host/network divide by thinking of networks as computer peripherals and designing a separation of host and network concerns within the kernel, and people thinking of them in this manner perpetuates the divide.

A crucial first step to bridging the host/network divide is for security practitioners starting to think of their world as a logical network of communicating processes rather than a physical network of communicating machines. As Sun's maxim goes, "the network *is* the computer™" [1]. My work is like a catwalk across the host/network divide. A production-strength bridge must supersede the catwalk using better standard kernel mechanisms; then we can overcome the human dimensions of the divide with new tools and visualizations.

The Need for Communication Visualization

My discussion with system administrators regarding security incidents and common problems gave rise to several usage scenarios where communication visualization would be beneficial. For example:

- **Detecting Covert Communications**: When we see network traffic for TCP port 80, we often assume that it is web-related, but with packet-process correlation and visualization, we could see whether the clients and servers are really web browsers and web servers.
- **Spyware and Adware**: Rather than running tedious system scans to locate spyware, a user could have ambient awareness of his machine's network activities and communicating processes.
- **Fine Tuning Firewalls**: Testing for vulnerabilities becomes more complex when one or more firewalls are between the tester and the target host. Being able to observe the effects of traffic on the target host with a host/network bridge installed would be helpful.
- **Cluster Computing**: Administrators who maintain large cluster computers could use this view to visualize communicating processes in the cluster and monitor for malicious activity.

In each case, visualization of packet-process correlation would benefit the administrator and user by complementing existing tools and enabling quicker diagnosis of problems. Even automated intrusion prevention systems could make better decisions if they could take the process names into account.

1.2 Research Questions

Having worked in computer security for several years prior to undertaking this research, I was aware of the pressing need system administrators had for good tools to comprehend the tremendous amount of data they see every day. Logs are helpful, but with thousands, perhaps millions of lines of logs to review daily (or even hourly), most system administrators are unable to cope. Additionally, I found that out of humility, caution, or simple uncertainty most system administrators were hesitant to declare their systems secure. Secure systems are important for all users of computers, and system administrators are critical to the security of every computer system. Automated security programs can

be helpful, but the ultimate responsibility and decision-making power is in the hands of system administrators with all the advantages and shortcomings of their human state.

Thus, my first research question addressed finding what was missing from the system administrators' intrusion detection toolkit that could make their jobs easier. A cursory glance at the tools in use by system administrators around me showed that there was a wide variety of text-based tools, but very few graphical tools, especially in the free or low-cost price range. When I took a course on Information Visualization[7], I found what I believed was the answer for the system administrators' needs. My second research question then addressed how I could apply information visualization techniques to improve system administrators' tools. The third question is related to the others because I saw that many tools were designed with a problem in mind but without taking usability into account. The third question addressed how I could apply human-computer interaction (HCI) methods[8], especially usability engineering[9] to develop effective tools for my user community.

Particularly, I decided that bridging the host/network divide would be an important help to my user community. Bridging the divide raised some other more specific questions about the value of a bridging technology to my users. Assuming the divide was bridged from a technical standpoint, what would be the best way to present the bridging data to the user? If a visual bridge were employed, what information is most important to present, and what would the visualization look like? What kind of information filtering technology is needed to present a compact, understandable view of the data to my users? During what part of the intrusion detection process does bridging data help my users most: before an intrusion is suspected, during the investigation, or as a forensic tool?

To recap the questions:

1. What tools are missing for system administrators to quickly notice and accurately diagnose security problems? Why do these tools not already exist?
2. What view of the data is most important for security awareness? Can information visualization techniques help system administrators obtain this view?
3. How can I apply human-computer interaction methods and usability engineering to the problem of helping administrators detect potential intrusions more quickly?
4. Can we correlate processes with their network traffic by bridging the host-network gap?
5. Can a visualization of this correlated data provide more insight with greater accuracy than existing tools?
6. What investigative timeframe is this data most important to my users?

This dissertation presents my discoveries about the system administrator user community and how I went about answering these questions.

1.3 General Method and Solution Overview

My approach to answering these questions was to ask them of members of my user community, HCI experts, and other researchers. I conducted a series of initial interviews to clarify the problem, researched available solutions in the literature, found what was missing, and worked with users to build it. I employed participatory design[10] by building a series of prototypes of increasing fidelity and getting the users' comments, complaints, and recommendations on them. The system administrators became my co-designers as I built what I believe will address the problems I have identified.

I developed several of my prototypes to the point where they could be subjected to formal usability tests. I used these tests as a way to understand my users' desires more concretely. In each case, I recorded their positive and negative feedback and their suggestions for improvements. I asked them probing questions to determine how they would use the tools in their jobs and work environments so

that I could understand whether the tool I was developing would be truly useful to real administrators. Then I incorporated their feedback into the succeeding prototypes.

The solution I arrived at is a mixture of a visualization tool and an underlying support structure. I determined that the users' need to correlate network events to processes running on monitored hosts could not be met on any existing operating system without modifications to the kernel. So I made the kernel modifications and wrote a visualization of the correlated data.

1.4 Impact

My work has implications for the kernel structure of all operating system kernels that use a TCP/IP protocol stack and network model. I demonstrate how the separation of concerns engineered into the layered TCP/IP model actually prevents certain types of analysis. In fact, my work shows how important it is to think of networks of communicating processes as basics computational units, not the individual host computers. If operating systems designers accept my findings, newer versions of kernels will enable much better monitoring of process and network activities and thus safer systems and a safer network.

1.5 Structure of the Thesis

The second chapter presents how I discovered the problem I am addressing via ethnographic interviews with system administrators. In this chapter, I examine the security work system administrators do and the types of analysis they perform when searching out and investigating potential security breaches. I discuss the various problems faced by administrators and how software may be employed to solve those that can be solved by software. I conclude by presenting the key findings from my interviews.

In chapter three, I present my examination of the security awareness tool literature as I sought to find existing solutions to the problems system administrators reported in the interviews. I present a taxonomy of existing solutions that classifies all the literature I surveyed and shows where the unmet needs are. Then I discuss examples from the literature review and explain how they fit into the taxonomy and why they do not solve the problem I am seeking to address.

Chapter four outlines my journey of evolutionary prototyping to reach the final solution to the problems I discovered. I discuss the low-fidelity and high-fidelity prototypes of import along the way and how each contributed to a better understanding of my users and their needs. For the high-fidelity prototypes, I discuss the formative usability evaluations I conducted and the lessons I learned from each.

In chapter five, I present my final solution for visually correlating packets on the network to processes on the monitored hosts, the Host-Network (HoNe) bridge. I start by illustrating how an intrusion may be detected using the HoNe visualizer and then discuss how this solution is superior to previous prototypes. I present a description of my modifications to the kernel and the reasons why they are necessary. I explain each part of the infrastructure and present an overview of the visualization.

The sixth chapter describes the summative evaluation of HoNe and its results. I discuss the pilot study and its results and the quantitative study from experiment design through the statistical analysis of the results. I conclude with a summary of the qualitative findings from the experiment that provide perhaps more information than the quantitative results that precede them.

The final chapter concludes the study with suggestions for future work and a recapitulation of the contributions my research offers the community.

CHAPTER 2 DISCOVERING THE PROBLEM

Having worked at a U.S. Navy lab during my first career, I knew that system administrators had a tremendous amount of data to sift through. At Virginia Tech alone, an estimated seven terabytes of packet data crosses our networks every day. If this data were printed out 66 lines and 80 columns to a page, double-sided, the daily stack would be 42 miles high.

After attending a class on Information Visualization, I became convinced that this was the method needed to compact this huge amount of data into something system administrators could use. A research assistant and I interviewed 24 system administrators (selected by recommendation and happenstance) from two universities to determine if information visualization approaches could help them ensure the security of their systems and networks.

2.1 Problem Scenarios

This section contains four common system administration situations where bridging the divide would provide critical insight to speed the task of understanding and resolving problems on networked hosts. Most of these situations were first presented to me during my interviews with system administrators, discussed in detail in section 2.1.

First, the increasing menaces of spyware and adware could be detected more easily with a view that bridges the host/network divide. Programs like netstat and lsof that do not collect traffic but simply poll kernel data structures are inadequate for detecting covert channels used by spyware and rootkits, because such communications can be set up and torn down within milliseconds. But bridging the host/network divide from within the kernel does not result in either large amounts of generated data or a heavy burden on the processor, and it gets *every* packet.

Next, consider the problem of detecting a kernel-level covert communications channel (perhaps installed as part of a rootkit). The rootkit makes the compromised machine's operating system lie to the user to protect the intrusion. An attacker can hide files, processes, disk space, and communications from even close examination. However, for a compromised host to be useful to an attacker, it must communicate. The key to detecting such intrusions is to compare communications activity seen on the network with process activity on the suspected host.

Another scenario involves fine-tuning firewalls. As "personal firewalls" become more widely deployed, penetration testing becomes more complex. A tester scanning a web server for vulnerabilities must ensure that her own firewall's inbound and outbound filters do not prevent her from seeing the results of the scan. With potentially many firewalls and other filtering devices separating a remote administrator from the device she is testing, she becomes less certain of the accuracy of her scans. A better approach would be to observe traffic simultaneously from several vantage points and correlate it to both the scanner process's socket use and process activity on the system being scanned.

A final compelling case concerns cluster-computers with a large number of similarly configured and jointly administered component computers. The inherent homogeneity of a cluster increases the risks of cascading intrusions. Currently, there is no good way to monitor the total security state of a cluster (although work is in progress[11]). An administrator must examine logs and traffic data looking for unusual patterns. If he finds something suspicious, he must then switch tools and look inside the suspect host(s) to find the problem. It would be better and faster if he could view the communicating processes in the context of the network traffic patterns that first led him to suspect a problem.

In each scenario, my bridge across the host/network divide can help the administrator solve problems more efficiently. Furthermore, using a visualization of data going across the bridge could

provide ambient awareness of communications so users would know at a glance to whom their computers were talking. Such a tool would complement existing tools, enabling quicker diagnosis of suspicious events. Additionally, automated systems such as Intrusion Detection Systems (IDS) could use the leverage provided by my bridge to gain greater power for diagnosing the causes of network and host events.

2.2 Why employ a visualization?

Bridging the host/network divide via new kernel mechanisms provides useful information to analysts hampered by tools that can view only one side or the other. But what is the best way to present this correlated information to the user? It is important not to confound the idea of bridging the divide with that of presenting the data. I believe that a visualization is a very beneficial way to present the data that the bridge makes available. This section discusses some of the benefits that a visualization of this data may provide.

One of the hardest parts of securing a set of networked computers is constructing an accurate mental model of what is happening so that appropriate action can be taken. Text data is absorbed sequentially via the auditory cognitive modality[12] as is speech, but graphical data can take advantage of the parallel nature of the visual/spatial modality. Thus I hypothesize that by visualizing packet traces, network administrators can more quickly and efficiently identify communication patterns in their networks.

Visualizations enable people to quickly identify gross patterns despite complex relationships in the underlying data. When a suspicious pattern appears, users may employ traditional tools to investigate further. Administrators can use my visualization to rapidly uncover suspicious activity on a machine. A simple visualization of machine and process icons joined by communication lines can show a wide variety of important patterns that may have security implications. Such a visualization shows a host view of communications coupled with the network view. Table 1 lists the cases communications can be classified as and the alternative benign and malicious diagnoses each case indicates.

Table 1: Potential diagnoses that visual communication patterns may indicate. Diagnoses of malicious behavior appear in boldface.

		Network View		
		Host is source	Host is destination	Host is listening
Host View	Associated with a normal process	• Normal client application connections • **Virus infecting a normal application**	• Connections to listening host server process • **Established** normal or **malicious connections**	• Host is running a sniffer, an intrusion detection system, or some other monitoring software.
	Associated with no known process	• **Covert channel traffic** • **Spoofed traffic** • ICMP response messages	• Misdirected or rejected traffic • **Attack traffic** • ICMP messages	• Host is misconfigured • **Host is running a covert sniffer**

From the point of view of the monitored host's network stack, the host is usually either the source or destination of each packet it sees. The host may also be simply listening without being part of the conversation. Traffic whose source or destination is a process on the monitored host is malicious if and only if the process is malicious. Traffic destined for the host but not traceable to any process may have been sent in error or may indicate an attack. TCP or UDP traffic that appears to come from the host but cannot be traced to any process is almost certainly malicious. If a host listens to packets that correspond to none of its processes, it may have a misconfigured network interface running in promiscuous mode, or the host may have a malicious covert listener installed.

2.3 System Administrator Interviews

I began this research with some knowledge of the vast amounts of log data that system administrators must examine each day and tried to devise information visualizations that would help them sort out this information more quickly. I soon discovered that I did not have sufficient understanding of how this information was being used in practice to proceed. I did not know how much time examining this data takes on the job nor what tools system administrators used to cope. I decided to ask system administrators directly.

I used a semi-structured interview protocol that covered the same general topics with each subject but did not always ask identical questions of everyone. Thus my results from this initial study yielded insights that are probably accurate but are not easily quantifiable.

Four interviewees were my pilot group who helped me develop the interview protocol I used to interview the other twenty. Eleven of the interviewees also helped me with expert reviews of prototypes I developed. The interview protocol included biographical information, security-related duties, intrusion detection experience, tools used in security monitoring, and breakdowns that plagued organizational security incident processes. An ethnographic summary of the interviewees appears in Table 2.

Table 2: System Administrator Ethnographic Summary

Area (# Subj)	Job Description	Scope of Responsibilities
Servers (9)	Manages web, file, mail, compute, etc. servers with little or no user contact.	Average of 23 servers and 3 users.
Users (7)	Manages end-user workstations for individuals or labs; lots of user contact.	Average of 4 servers and 75 users.
Security (2)	Receives data from other administrators, sets security policy, coordinates response to security incidents, forensic investigation, risk analysis, law enforcement liaison.	Indirectly responsible for entire university estimated at 300 servers and 40,000 users.
Network (2)	Monitors health and welfare of enterprise network; investigates ways to improve performance.	Responsible for network infra–structure, usage, and planning.

2.4 System Administrator Ethnographics

I set out to interview system administrators naively thinking of them as a homogenous group. Instead I found at least four distinct job types in the group of subjects studied:

1. System Administrator for Users (SAU): Manages end-user workstations for individuals or labs; lots of user contact.
2. System Administrator for Servers (SAS): Manages web, file, mail, compute, etc. servers with little or no user contact.
3. Security Officer (SO): Receives data from SAU/SAS, sets security policy, coordinates response to security incidents, forensic investigation, risk analysis, law enforcement liaison.
4. Network Analyst or Researcher (NAR): Monitors health and welfare of enterprise network; investigates ways to improve performance.

All of the interview subjects had some SAU/SAS experience, with a median professional experience of 14 years (mean 13.39, stdev 5.25). The distribution was heavily biased toward highly experienced administrators with only about one-third of the subjects having fewer than the mean years experience.

I concentrated the interviews on how the subjects discovered intrusions in their systems and what tools they employed to look for intrusions. A secondary focus was on obtaining ideas for and evaluating prototype tools for communications visualization.

The interviews were semi-structured, mostly lasting between 45 minutes and one and one-half hours. Although I used a list of questions, I tried to use them as springboards for conversation rather than a strict agenda. The advantage of this approach was that I learned many useful things that I did not know to ask before. Although after several interviews, I began to hear similar themes, there was not a single interview where I did not learn some valuable new concepts pertaining to the particular expertise of the interviewee. The disadvantage of this approach is that apart from the biographical questions (which I generally adhered to completely) I did not ask all the subjects the same questions, and was thus unable to draw quantitative conclusions from the data collected.

I found that the system administrators I interviewed had two main types of security-related activities that I've designated *Administrative Security* and *Pure Security*. The following is a breakdown of the types of duties involved in each type of activity.

1) Administration
   a) Patching software
   b) Managing users (education, individual service, problem-solving, etc.)
   c) Maintenance (log file reading, adjusting configurations, etc.)
   d) General network awareness (informational)
2) Pure security
   a) Staying informed of exploits, etc.
   b) Forensics (post-intrusion host analysis)
   c) Response and recovery
   d) Network investigation

After discovering the types of administrators and activities, I associated the types of activities to the administrator job descriptions to get a more complete view of how various types of administrators spend their time. This information is summarized in Table 3.

Table 3: How administrators spend their time.

	Activities	Job Descriptions SAU	SAS	SO	NAR
Admin-istration	Patching software	+	+	-	-
	Managing users	+	-	-	-
	Maintenance	+	+	-	0
	General network awareness (informational)	-	0	0	+
Pure Security	Staying informed of exploits, etc.	-	-	0	0
	Forensics	-	0	+	0
	Response and recovery	+	+	+	0
	Network investigation	-	-	+	+

'+' = a major activity of this job
'0' = a normal part of the job
'-' = a minor duty or not a duty at all

I constrained each job description to an equal number of '+' and '-' entries because all full-time jobs have the same available number of hours in the week, and it would be misleading to say that everything is a priority for some job. This way we can compare the job descriptions more clearly. Note that the above are rough estimates based on my understanding of the jobs, not a quantitative finding.

2.5 Findings

Less than a quarter of the subjects had any recollection of a security incident on machines they manage directly although all of them had knowledge of at least one major incident on a system someone else owned. About half of these were anecdotal recollections while the other half were actual experience helping clean up the affected systems. Many subjects were not able to quantify the percentage of their job spent on security-related activities. Of those who were able, the typical estimate was 20% of their time. Notably, several subjects said that security was either an additional duty or should take only 10% of their time according to their job description. From this I infer that my premise (that security does take too much time) may be accurate.

About a third of the subjects used some form of host-based intrusion detection system (IDS) regularly (especially Tripwire file integrity checker[13], and PortSentry[14]). Only two of the subjects had any experience running a network IDS such as Snort, and only one ran it regularly. The primary reason for the lack of network IDS was that the network management and security was often outsourced.

The vast majority of malicious activities witnessed by the interview subjects were "script kiddies" scanning for specific vulnerabilities or attempting to launch exploits. Since these attackers are not sophisticated, they often don't even attempt to do complete reconnaissance first, they just spring whatever exploit they have immediately. Port scanning and network mapping comprise the second most frequent suspicious type of activity but these are hard to separate out from the "script kiddie" activity. All system administrators agreed that the serious, talented "black hat" hackers are very rare and nearly impossible to detect. As a result, all but a very few of the subjects concentrate their efforts solely on two predominant threats. One administrator observed that if a black hat hacker has broken in

and he's not doing anything, it won't be detected, and it will probably cause little trouble. But the more he does on a system, the more fingerprints he'll leave, and the more likely he will be caught.

2.5.1 Detection and Diagnosis Methods

2.5.1.1 Detection Methods

The Grapevine: I found it interesting that the vast majority of the subjects reported that they discovered intrusions most often through user or external administrator complaints. I expected that intrusion detection systems or vigilant watching of the log files would find most intrusions. Instead more than three-quarters of the subjects cited user complaints about system performance and complaints from external system administrators about the behavior of their systems as the primary way they discovered intrusions. The administrators interviewed said users who uncovered an intrusion typically first noticed slow or erratic system behavior, sudden reduction of free disk space, error messages, system crashes, or defaced websites. Since most of the interview subjects reported regularly monitoring log files, etc., I surmised that there is a problem with existing tools that allows administrators to miss important evidence of intrusion activity. However, with semi-structured interviews, I was unable to obtain better than anecdotal-quality evidence of this perceived problem.

Logging System Anomalies: Less commonly, subjects discovered intrusions via evidence in the log files. Odd communication patterns (scans from within the home network, unauthorized use of well-known ports or use of forbidden ports, blacklisted IP source addresses, or unexplained spikes in network usage) were the most common indication of potentially dangerous activity found in log files. It is worth noting that "odd" presupposes the ability to recognize normality. Being able to distinguish the odd from the normal is a skill that none of the subjects could concretely explain apart from their experience. System administrators reported that log files (at least on Unix-based systems) do contain understandable information that can be used to uncover intrusions, but they also said that there was too much information in the logs, making the process harder. System administrators who did not yet have a log-reducing service running said installing one was a near-term priority.

System administrators regarded Windows system log information to be much less useful than Unix-style log files. Key problems with Windows logging were (1) that domain and machine names have no outside registration and can be easily spoofed, and (2) that much of the information contained in the error log was encoded as an integer error number without accompanying text. As far as the subjects knew, these error numbers seemed to be assigned randomly to conditions, and they knew of no official reference list that contained the complete set of error numbers and their descriptions. These defects probably contributed heavily to my observation that logging system anomalies were not as frequent a detection method as external complaints.

File System Anomalies were the next most commonly reported as ways the subjects noticed intrusions without help from another person. Some examples of file system anomalies are: deleted logs, apparently deleted sections or types of logs, inability to log, presence of unusual files and directories, and changed configuration files. Again, to determine whether something was anomalous or not required detailed knowledge of what the file system should look like under normal conditions.

Behavioral Anomalies: Subjects cited machine behavioral anomalies (improper function of common commands, spontaneous rebooting, etc.) as another indication that an intrusion may have occurred. Several subjects reported that they noticed behavioral anomalies via a "sixth sense" indication that the machine was not operating properly. I believe that the need for intuition about system behavior indicates that much of system administration is still a "black art" where competence is only developed via years of experience.

Automated Notification: In all the experiences reported by all the subjects, only twice was near real-time notification of problems from an automated monitoring system given as the way a system compromise was discovered. Both times the Big Brother[15] monitoring system first noted the problem. Very few of the subjects used any type of network intrusion detection system (NIDS), but this was probably because for most of the subjects, network management was outsourced to a competent external agency. Nearly half used host-based intrusion detection such as Tripwire[13], and most used host-based firewalls such as IP-Tables (IP-Chains)[16] and ZoneAlarm[17]. However, none of the interviewees used NIDS. The reasons I heard cited most often for not running NIDS were: insufficient time/resources to install, configure, and monitor NIDS, and the problem of false positives. Subjects said that cryptographic checksum comparison warnings from Tripwire[13] were the most common way to detect modified files. Despite the lack of NIDS, all the subjects were fairly confident in the security of their machines mostly due to human vigilance.

One subject was notified of a new vulnerability from his system security officer and upon investigation, discovered that some of his systems had the vulnerability and had been exploited. This case was a day-zero exploit[1] and many similar systems were affected. The subject in question was coincidently not harmed by the exploit because it installed a sniffer on an unused interface. However, other similar systems across his organization did suffer compromised password and data. Although this method of detection was quite rare among interview subjects, it does underscore the importance of interpersonal communication between security officers and system administrators. One could argue that this demonstrates the importance of a separate security officer position whose responsibility it is to monitor for new exploits and be aware of potential vulnerabilities within the organization.

2.5.1.2 Analysis Methods

Although few system administrators admitted to having had intrusions on machines under their purview, almost all of them had some experience diagnosing and recovering from successful intrusions. Most of the experience cited was helping other system owners recover compromised machines. Several of the interview subjects belong to a Computer Emergency Response Team (CERT) and part of their unofficial responsibilities were to assist when compromises were found.

I identified three kinds of analysis that subjects used to detect security problems, each at a different time relative to an incident:

1) Informative: When the administrator has no suspicion of malicious activity, she may use tools to periodically check the security state of her machines.
2) Investigative: When the administrator suspects malicious activity and is seeking to confirm it (e.g., after an IDS alert), she may use tools to gain a mental picture of the overall situation to focus further detailed search.
3) Forensic: When the user has confirmed the malicious activity and is seeking to locate the processes, files, etc., responsible for the behavior.

Informative monitoring is performed routinely on (assumedly) uncompromised machines to discover potential security problems. Investigative analysis involves diagnosing problems on a suspected compromised system running in its normal environment. Forensic analysis is usually done with a compromised system offline. While some types of exploits are only visible when the host is active and connected to a network, many exploits are so harmful, or their victims are so sensitive, that the compromised machine must be taken offline for examination.

[1] An exploit that takes advantage of a vulnerability on the same day or before the vulnerability is generally known

Interview subjects reported many types of analysis that I have grouped into four categories based on what data is being viewed:

1) Process Analysis
2) File System Analysis
3) Activity (log) Analysis
4) Vulnerability Analysis

Process Analysis: The most commonly cited analysis type used for investigative assessment was to view the processes running on the host looking for unusual names or numbers of processes. Of course, many rootkits automatically replace the process-viewing tools (e.g., Unix ps command, or Windows Task Manager) with infected versions, but consulting these tools first is a relatively low-cost check.

Several administrators investigated unusual process names via an Internet search engine (such as Google.com) or an online vulnerability database. Surprisingly, subjects considered search engines to be faster and more accurate than vulnerability databases for quick lookup of information on compromises. Subjects also used the /proc file system (available on Linux and some other forms of Unix) to inspect processes. The /proc file system is a run-time reflection of the execution state of the computer with directories for each running process and files for each environment variable used.

File System Analysis: Analysis by examining the file systems for unusual or modified files, especially executables, dynamically linked libraries, kernel modules, and service configuration files is more intensive than process analysis. This kind of analysis requires significantly greater expertise to "know what to look for" and make a diagnosis. File system analysis is thus not often used unless there is strong suspicion of a compromise.

Many tools perform limited forms of file system analysis, particularly virus scanners. Interview subjects commonly used several of these tools before attempting to manually diagnose file system problems. Interview subjects mentioned several file system intrusion indicators: nonstandard file systems, unusual files such as rootkits and hidden directories, changed or unusual executables, libraries or configuration files, presence of MP3 (music) or AVI (movie) files not installed by the machine owner, and deleted log directories and files. Subjects would also examine configuration files for correctness, but only when a particular service is suspected of compromise.

Activity (Log) Analysis: As with process analysis, this form of analysis looks at what the machine is doing, but it has an historical dimension that is gleaned from the information stored in the log files. Activity analysis implicitly includes the communication activities of the host. Using the log files, an experienced system administrator can piece together what may have happened to the system in the recent past. Some important kinds of log files for this use are: system log files (for recent logins, privilege use, etc.), service log files (e.g., Apache's access_log), error logs, history and cache files, intrusion detection and firewall logs, and packet traces. Not all activity analysis is done via log files. Some telltale activity, such as frequent disk accesses when the machine should be idle or wildly flashing communication lights, can be noticed just by physically observing the machine.

Some difficulties with activity analysis are multiple log file formats, poor documentation of log file formats, inconsistent timestamps between machines, high volume of information with low relevance, and the likelihood that standard log files and directories will be immediately deleted or modified upon successful intrusion. Some of the evidences of intrusion my subjects cited were: network activity on

unexpected ports, unexpected use of a protocol or port[2], exceptionally high traffic (greater than three standard deviations above normal levels) on any port, automated emulation of user interaction (e.g., attempted logins many times a second), and legitimate logins from unexpected locations or at unexpected times.

Vulnerability Analysis: Subjects looked at common or recent exploits and vulnerability reports and used these as heuristics to find intrusions. The rationale is that the most likely avenues of compromise are exploits that someone has recently seen. This may be a dangerous assumption to make given that worms such as Slapper[18] have reached their saturation point on the entire Internet within fifteen minutes. Old exploits still exist in the wild, but after a few months the new infection rate appears to be greatly reduced, but vulnerability analysis typically looks for these well-defined exploits first. New exploits may thus achieve tactical surprise. Next to process analysis, vulnerability analysis is the simplest, most economical diagnostic analysis. An advantage of this analysis is that it ensures that known problems are addressed first.

When we consider the times and types of analysis together, we can see when a particular analysis method is most and least applicable. In Table 4, I have depicted the relationship between time and type of diagnosis. Within each analysis time, I have ranked the importance of each type of analysis. The rankings are subjective based on my understanding of what the interview subjects were telling me. Taking the average rank of each type of analysis reveals that Activity and Process analysis are the overall most applicable types. To be most helpful to my users, I decided to concentrate on these two types of analysis.

Table 4: Relationship of times and types of diagnosis

		Analysis Time			
		Informative	**Investigative**	**Forensic**	**Average Rank**
Analysis Type	Process	3	**1**	3	**2.33**
	File System	4	3	**1**	2.67
	Activity (log)	2	2	2	**2**
	Vulnerability	**1**	4	4	3

Legend: **Most important**; Moderately Important; Least Important

2.5.2 What Keeps the Wily Hacker Out

This section discusses the factors the interview subjects attributed their success to. At the top of almost everyone's lists were vigilant patching and removal of unnecessary services. It has always been important to keep system software up-to-date, but in recent years this has become critical. Gone are the days when it was common for Internet hosts to provide open services to the rest of the world. Now the mindset is to minimize access by exposing to the Internet only those services that are absolutely required. Most subjects also separate exposed services to one per machine. A web server generally does not also provide other services such as e-mail. Typically the subjects interviewed would expose

[2] For example, http sessions that last for hours instead of seconds might indicate that port 80 is being used by an rogue process or that an intruder is tunneling another protocol through http.

only one other service apart from the primary service of the machine to the Internet, and that is secure shell (SSH) service for remote administration.

More than half of the interview subjects had outsourced network management and attributed much of their security to the effectiveness of the campus ISP's security. The campus ISP provides no enterprise-wide firewall but is very security conscious and monitors for intrusions constantly. Some subjects whose networks connect via this ISP were unhappy with the lack of a central firewall but most saw it as an advantage. The lack of a network firewall meant that all the machines had to be patched and host-based firewalls and IDS had to be deployed. A centrally managed network firewall requires coordination whenever a user needs to allow a new protocol in or out. Additionally, a central firewall is far more subject to insertion/evasion attacks[19] than host-based protections. In an academic institution (such as where many of the interviewees worked) research and freedom of speech make restricting traffic quite complex. The ISP chose to allow anything to come into the network but otherwise to secure the network by careful monitoring. The disadvantage to this approach is that if a system goes on the network unprotected, it is sure to be hacked. The ISP does not worry about individual machines that are hacked, but if they become a danger to others on the network or use too much bandwidth, the ISP does shut them down by disabling their network access at the Ethernet port (wall outlet) level.

System administrators I interviewed attributed some of their success to defense-in-depth (having multiple layers of security controls). Some common layers of security employed were host-based firewalls, host-based IDS[13][14], additional logging, maintaining hot-spare servers, and centralized identity management.

Interview subjects by-and-large reported that their management in recent years has become increasingly security aware and supportive of security measures. The causes of the security-friendly culture my subjects cited were increased negative visibility due to security breaches and awareness of the possibility of real monetary loss.

One of the subjects cited the ability to assume a "black hat" hacker's mindset when looking at a system's security measures as a reason why his systems had not been hacked. The ability to predict potential security holes of given features and implementations is generally valued in the system administration field, although administrators vary widely in ability or desire to assume this mindset.

At Virginia Tech, where most of the interview subjects worked, there are many ways that ideas are shared among system administrators and other computer support personnel. One way is a "support for support" mailing list where subscribers share ideas, problems, news, and opinions. Another is the Computer Incident Response Team (CIRT) composed of many of the members of the administration community. This team runs on the "volunteer fire department" model where everyone volunteers his or her time and helps out when a security incident requires extra hands and minds. The team members have a tacit agreement with their supervisors that they will be available as needed.

Subjects reported that certain system settings, policies, and configurations were responsible for a great deal of their success. First, the policy of not responding to Internet Control Message Protocol (ICMP) messages or responding to only a limited set proved very successful. ICMP is very easy to abuse and many enterprises simply disallow it at the firewall. Assailants use ICMP to map networks, either by presence or absence of the pinged addresses, so the interviewees found that making no acknowledgement was the best course of action. Subjects reported diminishing rates of hacking attempts after disallowing ICMP. A downside of this practice is that without ICMP, it is also difficult for the administrator to check whether particular machines are up or down, especially from a remote location.

For mail servers, disallowing or quarantining certain types of file attachments (e.g., .pif, .exe, etc.) prevented most types of viruses and worms from spreading through file vectors. However, as the MyDoom[20] variants showed, even "safe" attachments like .zip files can prove dangerous. Finally, configuring servers to run only a single service each (rather than for example running your web, mail, and file servers from a single machine) helped prevent problems. This is partly because the host-based firewall on the server can be much more specific in the ports allowed in or out of the server. Also, this allows administrators to establish expected patterns of traffic to and from their machines that can help them highlight anomalous behavior.

A few subjects cited the adequacy of freely available or low-cost administration tools as a reason for their success. Apparently, the organizations these subjects were from preferred not to spend money on administration and security software. One subject even reported that she believed her organization found additional manpower cheaper than commercially available administration tools. Whether this is because budgets are too small or the prices of these tools are too high is unknown.

Separation of responsibilities was also cited as a common reason for success. Having a dedicated, autonomous security officer who reported directly to CIO of the enterprise was listed as a successful policy. Subjects cited separation of network security responsibilities from host security responsibilities as another successful approach.

2.5.3 What Makes the System Administrator's Job Hard

Some of the bad news is simply the downside of design tradeoffs that enabled the good news. For example, separation of concerns among separate persons allows work to be done more efficiently but may be the cause of lack of communication. Setting up new machines on the network required input from both department system administrators and the ISP. Additionally, the many independent shops that were within a system administrator's network segment but not under his control caused problems. Since the administrator cannot dictate policy to these shops, security problems may abound. A related problem is lack of effective communication between peer administrators working on interdependent systems. It is quite possible for the actions of one person to unintentionally nullify the efforts of another to secure the system. The prime example of this is uncoordinated application of patches to interdependent systems. One organization that uses a network firewall related the coordination problems arising from ensuring that firewalls were as secure as possible but still allowed the proper functioning of critical systems protected by them.

Lack of uniform security standards across departments was another cause of miscommunication and failure of human processes. In some departments, independent shops were given free reign and more than enough freedom to endanger themselves and others. Other departments insisted on strict control and centralized administration. Differences such as these within a single organization are potentially dangerous, especially when a user moves out of a strict, centralized department to one without such structure and finds himself unprotected.

Interview subjects saw lack of security education and awareness as one of the largest weaknesses in their organizations. Both users and system administrators could be stricken with this deficiency. For example: users would open unprotected network shares with world read/write privileges; they would use weak passwords (especially on shared accounts where the password is made easy for everyone to remember) and they circumvented password policies in a number of ways. Users would install software that opens security holes on their work machines or would attach infected laptops, etc. in a secure network. A user's technical savvy did not prevent him from being largely ignorant of security needs. In fact, technically advanced users would often be more resistant to security requirements until they were "burned" by a costly security incident. On Unix systems, people running setuid scripts for

convenience instead of using sudo was a concern since the former do not produce log entries that could be traced later. A final educational or awareness weakness cited by my subjects was users being tricked into sharing passwords or opening dangerous e-mail attachments via social engineering schemes.

The interview subjects also cited numerous problems with patching their systems. Often there were too many patches and often these did not work or broke things that had been working. Certain vendors did not give complete information about what issues their patches fixed making administrators uncertain whether applying the patches was worth the grief. Typically, patches came out more slowly than the exploits they were designed to fix, and even if they were timely, administrators often did not have time to properly test the installation of patches before exploits hit. Most important, the interview subjects felt that the patching problems were themselves caused by vendors who released vulnerable products in the first place. Finally, some subjects were using certain vendor-specific system control software that was incompatible with operating system upgrades and patches. Thus, these administrators had to choose between security and mission-critical functionality.

The interview subjects said that finding enough time to properly secure their systems was very difficult. Partly they cited budgetary and lack of training problems, but the sheer amount of information they must process is staggering as well. Effort to support a larger number of systems where some tasks could be automated scales well, but the effort required to support growing numbers of users does not scale well. The rapid growth of the number of new exploits also requires a great deal of time of system administrators. But an ongoing concern of many subjects was that even with all the time and money they thought they needed, the professional "blackhat" hackers would still be extraordinarily difficult to catch. This latter issue forms a constant, low-level concern for most security-minded administrators.

2.5.4 Tools

During the interviews, I asked the system administrators to list the tools that they used to secure their systems, investigate potential intrusions, and recover from successful intrusions. I have listed these tools by category in Table 5 and included the primary times and types of analysis supported by the tools in each category.

The interviewees seemed to be relatively happy with the tools they currently had, but several said they would have liked to use more expensive analysis tools that were out of their budgetary reach. Of course, they did have problems with their tools. Typical problems subjects cited with existing tools were:

- They were hard to understand usually because they provided too much information that should have been aggregated.
- Some were difficult to configure and maintain.
- Commercial tools were too expensive for their organization while low-cost and freeware tools were inadequately supported by their authors.
- Most tools were incapable of real-time analysis either because they required too much preprocessed data or because execution time was too slow.
- Tools were often immature:
    - Log checking programs could not handle the many different log file formats
    - Require human intervention to validate data
    - Security tools often lacked quality control
    - Many tools work only according to the vendor's/writer's preconceptions of how they should be used rather than the users' concerns.
    - Coding errors in some tools can lead to false positives/negatives.

Table 5: Tools commonly used by interviewees for computer security needs

Tool Category	Examples	Analysis Supported
Log reduction programs	Logcheck, LogScan, LogWatch or whatever comes with the system; Swatch; Custom written scripts	• Informative activity analysis • Investigative activity analysis • Forensic activity analysis
Host-based IDS	Tripwire, PortSentry	• Informative file-system analysis • Forensic file-system analysis
Firewalls	IP Chains, IP Tables, Cisco PIX	• Informative activity analysis
Native OS tools	Task manager or ps command, netstat, lsof, iostat, vm_stat, pstat, top, kextstat, etc.	• Informative process and file system analysis • Investigative process and file system analysis • Forensic process and file system analysis
Network management tools	*Multi-Host-Oriented*: Big Brother; Nagios, Orcallator; eHealth; *Network-oriented*: Ethereal, Traceroute, TCPdump, TCPView, EtherMan/EtherApe; Multi-Router Traffic Grapher (MRTG)	• Investigative activity analysis
Vulnerability assessment tools	Nessus; Center for Internet Security benchmark and scoring tools. (www.cisecurity.com); Port scanners (nmap, etc.); Retina (eEye Digital Security)	• Informative vulnerability analysis
Online information sources	Google (for finding meaning of unusual process/file names); BugTraq, etc.; DShield	• Investigative process and activity analyses • Forensic process and activity analyses
Forensic tools	Coroner's Toolkit; In-house-developed, custom software; PC File Recovery: to check for recently deleted files; Application Mapper (THC-AMAP); Free tools from Foundstone;	• Forensic file-system and activity analysis
Patching and updating tools	Windows Update; Unix autoupdate scripts; other vendor updaters	• Informative vulnerability analysis
Virus checkers	Various free and commercial tools	• Informative vulnerability analysis

Because the interviewees expressed frustration with existing tools, I asked them what their wishes were for new tools. To determine whether information visualization tools would be useful for system

administrators, I extracted all the needs expressed in my interviews—50 needs in all. All but three of the interviewees expressed at least one need. Some needs were near duplicates, while others were amalgams of several separate needs. After preliminary analysis and classification of the needs, I derived 45 distinct need items. Most needs were only requested by one person (mean: 1.156, variance: 0.309), and each person expressed an average of 2.6 needs (variance 4.989). The large variances indicate a study with a bigger sample size is needed to verify my conclusions.

The weight assigned to each need expressed by a single individual was the inverse of the total number of needs that person expressed. If more than one person expressed the same need, I added the fractions. Thus if need *x* was expressed by two persons, one who expressed four needs and the other who expressed two needs, the weight given to need *x* was 1/4 + 1/2 = 3/4. I used this approach because one person expressed a very high number of separate needs (eight of them, placing him in the 99th percentile) most of which fell into the same category, skewing the results. The results match my intuitive sense of user needs from conducting the interviews. A treemap[21] displaying the weighted need scores is shown in Figure 2. In a treemap, the larger the area, the higher the score. The treemap also shows the hierarchy of needs by containing subordinate needs inside their parent categories.

Figure 2: Treemap of security needs expressed by the system administrators interviewed. Larger area implies more needs. The yellow boxes are the areas I decided to concentrate on.

I found that two-thirds of the needs expressed could be met via software solutions. Awareness and visualization software comprised 45% of all needs, with about 14% being for network security, 21% for host security, and 7% for general visualization needs such as viewing large datasets. The two yellow areas in the figure above, communications awareness and large dataset viewing, are the areas I chose to concentrate my efforts on. I asked my interview subjects whether visualization technology

would be helpful to them, and most were enthusiastic about the prospect of using a visualization to compactly represent the large amounts of data that they needed to sift through.

Several of the interview subjects had ideas and requests for any new visualization technology. These suggestions I have categorized as: tool use ideas, visualization ideas, and critiques of paper visualization prototypes. Subjects said that a good tool would gather and integrate information from multiple, decentralized sources and present them visually. They said that the tools should make intrusion detection decisions based on as much as a few months' worth of data and that millisecond resolution was required for some kinds of diagnosis. These complementary needs implied gathering, storing, and indexing a tremendous amount of information. Thus subjects suggested that the tool keep a diminishing level of detail as the information grew older. Every event would be kept for recent hours of data, with samples and statistical summaries for older data.

Some of the suggestions for visualization would have made the new tool mimic existing tools rather than covering unique territory. Subjects wanted to be able to visually identify spikes of traffic of various types as seen on the network. Plotting over time would allow users to see the story as it unfolds. Because of the nature of network attacks, subjects suggested aggregating the data. This would give disproportionately large attention to smaller features. Subjects were afraid that a high level view of the data might hide small, unusual traffic such as ephemeral UDP that might be used for covert communication. However, one subject suggested that the bursty nature of network traffic would make exponential smoothing necessary.

Traffic visualization with the ability to filter out all common traffic was an important ability several subjects wanted in a new tool. Although the term was not mentioned, subjects indicated that a novelty detection approach would be useful. Thus, they would use the visualization to see anything that they could not describe with filters as normal. Subjects wanted the visualization to be interpretable at a glance. Administrators wanted to be able to see all the data unless they had specifically filtered it out. Creating an effective set of filters was considered a potentially difficult and complex problem requiring significant experience.

Some comments critiqued my paper prototypes of visualizations that I showed administrators as I learned what they wanted. Initially, I had designed a prototype that grouped hosts seen on the network into five discrete trust levels. However, subjects consistently said that they were initially interested only in whether the traffic source is from inside or outside their own network. They preferred to drill down to find out more details of individual hosts of interest. Subjects also said that organizing hosts into five trust levels or even into a blacklist would involve significant effort.

2.5.5 Interview Conclusions

My interviews demonstrated that system administrators are not as homogeneous a group as I had initially thought. I found large differences in duties, tools, and techniques between those who primarily administrate servers and those whose main job is to assist users. Security officers, network analysts, and operations center specialists had many duties in common with other types of system administrators, but they had different areas of emphasis and different tool support needs.

I was surprised that most of the interviewees preferred text-based tools to visualizations and other higher-end tools. This preference may reflect a population tendency toward low-level data analysis, but subsequent studies (especially [22]) have demonstrated that my user community actually prefers visual solutions but has not yet found satisfactory ones.

My interviews with system administrators indicated that the single most important indicator of intrusions was aberrant communication patterns seen on the network or in host log files (activity

analysis). The second largest indicator of problems was the appearance of suspicious processes on a host machine (process analysis).

Key Conclusion: In my interviews and subsequent studies, system administrators related the painstaking process they used to track down potential security incidents: They construct a mental image by fusing information from the network (activity analysis) and the suspect hosts (process analysis) into a correlated view of communications. However, the interview subjects knew of no tool to construct this view automatically. They were forced to do the analyses manually.

CHAPTER 3 EXAMINING EXISTING SOLUTIONS

When we consider that every packet on the Internet comes from some computer process on a machine that may be anywhere in the world, we can see the staggering amount of effort required to monitor these activities for security. Most administrators I have interviewed care only about a few hundred machines they are responsible to maintain. But even with relatively few machines, having to examine anomalous events from one side of the host/network divide at a time can be time-consuming and error prone. When an anomalous event occurs, system administrators must either investigate it using existing tools or write new ones. I will cover the writing of new tools later, but all of the existing tools incorporate the same division between host and network. They can look at one or the other, but the tools provide no correlation that maps packets to processes and vice-versa. The situation is pervasive, so people may believe there is no alternative but to consider the host and network separately.

Programs like Zone Lab's excellent firewall, ZoneAlarm[17], can tell a user which process a packet is emanating from on Windows™ machines, but it does not enable the same visibility from the network side. ZoneAlarm is more powerful than netstat because it enables the user to control connections. However, ZoneAlarm provides no visualization, nor can it provide remote monitoring of another machine. Foundstone's tools Fport and Vision[23] map open network ports to the applications, services, and drivers that opened them, but they cannot trace packets across the network, nor can they show this information about the open ports from the network side of the host/network divide. In contrast, tcpdump and other programs based on network sniffers, can catch and record every packet that a machine sees on the network but cannot tell the user whether the packets were seen or used by processes running on the receiving machine.

Netstat, lsof[24], and related tools are based on polling kernel data structures listing active processes, sockets, etc. The problem with polling is that a malicious process could open up a socket at random intervals to transmit a few UDP datagrams and then close it within a few milliseconds. It is nearly impossible for polling programs to identify the offending process, since the socket is gone before it can be traced. With existing tools, the work of bridging the host/network divide must done in the mind of the user.

No other known tool (freeware, academic, or commercial) at the time of this writing correlates network packets to the machine processes that generate it and visualizes the result. Some may integrate host information from multiple hosts on a network, and others may present network activity side-by-side with selected data from host logs, but none actually correlate each packet to a process on the machine that sent or received it and provide a visual presentation of this correlation. Thus, my approach is unique.

3.1 A Taxonomy of Available Tools

From my review of the tools available in practice and in the literature, I constructed a taxonomy of known tools organized by the context of the data they present and the way they present it. Figure 3 illustrates this taxonomy that I use to compare tools throughout this document. The fraction in the lower right corner of each category in the figure shows the number of interviewed system administrators who mentioned using a tool from that category. Representatives are known to exist at this writing for most of the possible types of security awareness tools.

Although there are many other ways one could classify security awareness tools, this taxonomy clearly shows what is missing from the system administrator's tool chest. The HoNe Visualizer is the only known visualization of the end-to-end view and the only networked host visualization that is suited to security requirements.

3.1.1 Communication Contexts

The abscissa (x-axis) of the taxonomy diagram has four views of communication context that security awareness tools typically employ: internal host, networked host, network, and end-to-end. The primary determinant for categorizing a tool is the information sources it uses. These categories proceed loosely according to the breadth of awareness the tool has about the meaning of computer communications.

An internal host view (IH) presents data that concerns only the monitored host(s) without regard to network connections. Internal data includes service and application log files (excluding network activity), process activity, memory and CPU utilization, etc. Generally, the data is interpreted only with respect to the reporting host. An internal view may be remotely reported, but it never shows network activity except as it directly affects the reporting host. Some internal views aggregate reports from more than one host, but they do not include traffic data from a network.

A networked host view (NH) displays data that concerns only the monitored machines, but includes the broader context of their network connections. Thus a tool of this type might monitor the network interfaces of a host and make the user aware of network connections involving that machine. The scope of analysis may concern users, applications, processes, sockets, ports, and network interfaces, but it does not enable examination of network traffic that does not involve the monitored host.

A network view (NW) presents traffic data in the context of a network or internetwork. Traffic data may include packet traces, network flows, timing and congestion information, and routing information. Network views may aggregate information from sensors on different networks, but they always interpret the data in a network-oriented context. Network views do not use any host data to interpret the meaning of network traffic.

An end-to-end view (EE) presents entire communications by interpreting process communication data on a networked host in the larger context of the network or internetwork where it resides. The basic unit of analysis in an end-to-end view is a communication, including the processes responsible for maintaining the communication on at least one end point and the traffic that carries the communication across the network(s) to the destination machine(s). End-to-end views may show more than one complete communication at a time.

Each of these communication context views is important, and none can substitute for the others. However, the networked host and the end-to-end views are seriously under-represented in the literature because these views require changes to the communication stacks in the operating system's kernel to allow freer communication between the network and transport layers.

3.1.2 Presentation Modes

The ordinate (y-axis) of the taxonomy diagram contains four ways security awareness tools may display information: Text-based, dashboards, summary charts and graphs, and visualizations. The general idea is that some kinds of displays are perceived by humans more linearly (sequential, audio modality) and others more spatially (parallel, visual modality)[12]. Another factor that comes into play here is the presentation's level of abstraction.

Text-based displays (TB) may have graphical user interfaces, but the information presented is pure text. Formatting may improve readability, but the user must still process the information sequentially. Similarly, dashboard-style displays (DA) present mostly text data, but use simple preattentive features (e.g., color and motion (blinking)) to draw the user's attention to critical items. Both text and

dashboard display types cannot easily compress large amounts of data into a single screen overview. Rather, the user must scroll through windows full of low-density, symbolic information.

In contrast, summary charts and graphs (CG) present abstract quantities such as throughput or inter-arrival rates. These presentations separate the quantities measured from their sources. Thus while charts and graphs make it easier to see overall trends, they also make it difficult to associate quantities with concrete entities or to drill down to underlying data.

The final presentation category, visualizations (VZ), alleviates this difficulty somewhat by reducing the level of abstraction. Thus, a visualization tool will generally not display an abstract quantity like latency, but will instead convey the state of some object (a machine, a service, or an alert, for example) via an abstract marker or icon.

When comparing visualizations, considering the degree of preprocessing (aggregation, interpretation, etc.) that the data to be visualized undergoes prior to presentation is advisable. This degree bounds the degree of potential drill down. For example: Packet header visualizations generally operate directly on raw packet data obtained by sniffing. Thus, drill-down to the packet field level is theoretically possible. In contrast, visualizations of intrusion detection system (IDS) data present preprocessed data correlated from one or more IDS. Typically, this data separates the packet(s) that triggered the alert from its context in the communications stream.

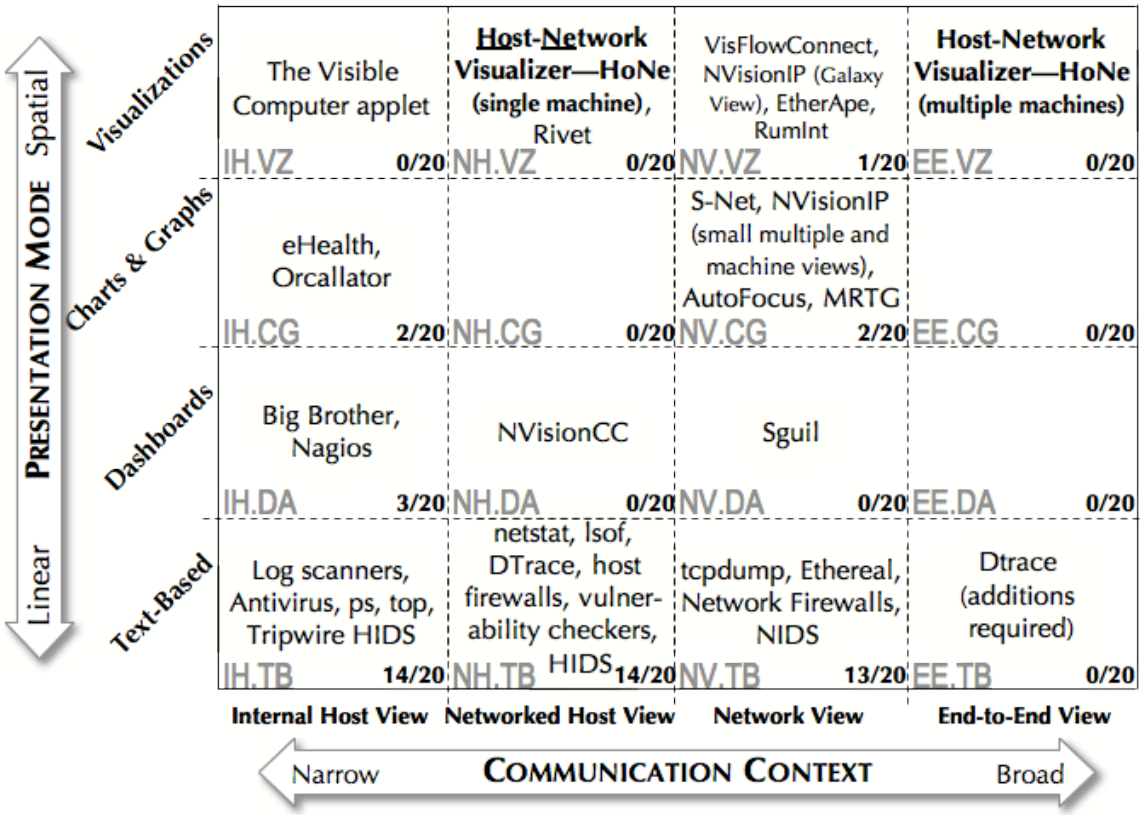


Figure 3: Taxonomy of security awareness tools. The lower left corner of each box is the taxonomic name of that class of tools. The lower right corner shows the number of interviewed administrators who regularly used a tool in this class.

Card, et al.[25] discuss the importance of visualizations for rapid, accurate, and intuitive understanding of large amounts of data. Good visualizations provide external cognition aids,

enhancing understanding and supporting human decision-making. Unlike sequential text, visualizations are perceived in parallel, using the faster visual modality rather than the slower auditory modality. Thus, I believe visualizations are superior to text-based approaches when large amounts of data must be understood and acted upon quickly. However, textual information is unsurpassed for gaining a detailed understanding of a small amount of data. Consequently, I believe that good visualizations should enable the user to drill-down to the underlying textual data when details are needed.

3.1.3 Taxonomy Abbreviations

Because some tools may have parts that fit into several different categories, I have established an abbreviated naming scheme. The abbreviated classification of any tool is given as a comma-separated list of abbreviated communications contexts, followed by a period, and finally the comma-separated list of abbreviated presentation categories. Thus I describe NVisionIP, which provides a visualization and summary charts and graphs of the network view, as "NW.VZ,CG: NVisionIP." Alternatively, I could say, "NW.VZ, NW.CG: NVisionIP" indicating that the tool has separate components or views that fit in two of the sixteen categories.

3.2 How the Literature Fits into the Taxonomy

In this section, I classify the known tools and approaches in the literature and in practice according to the taxonomy discussed above. I have ordered the discussion of literature first by communication context and then by presentation mode.

3.2.1 Internal Host Views (IH)

The majority (14 out of 20) interviewees reported using internal host view tools. This category of tools is fundamental to security monitoring.

3.2.1.1 Text-Based (IH.TB)

Text-based internal host view tools had the widest usage of any category of tools. All interviewees who used an internal host view tool used at least one text-based tool of this sort. This category includes some host-based intrusion detection systems (HIDS), antivirus packages, log scanners, forensic toolkits, security benchmarking tools, and many operating system utilities that provide process activity reports.

One example of IH.TB tools is the Center for Internet Security (CIS)[26] benchmarks. CIS has compiled lists of best security practices for several popular operating systems and encoded them as a series of tools that examine a machine's compliance with their recommendations. Another example is the Tripwire[13] HIDS. Tripwire uses a baseline of cryptographic hashes of critical files to determine when changes are made to a system. One quarter of the subjects interviewed used Tripwire regularly. Tripwire is another after-the-fact, periodic, security awareness tool. Because it reports changes to each affected file, Tripwire facilitates recovery from an intrusion.

3.2.1.2 Dashboard (IH.DA)

Only 3 of the 14 internal host view tool users mentioning a dashboard internal host views tool. The reasons for this unpopularity probably have to do with the fact that most tools in this category were for monitoring multiple hosts; thus the setup was likely to require both client and server installation and configuration. This category includes network node management tools like HP OpenView[27], Big Brother[13], Nagios[28] (see Figure 4), and What's Up Gold[29]. Some of these tools are capable of limited graphs and even visualizations, but their primary use in practice is as a dashboard. Since these are technologically very similar, I discuss them in general terms.

Network node management tools generally gather internal host data from multiple machines distributed across the network. Some of the hosts may be network infrastructure elements such as routers, but the data presented comes from internal host conditions, not network traffic, so they should not be confused with network views. Gathering may be done via Simple Network Management Protocol (SNMP) traps, Internet Control Message Protocol (ICMP) or other probes, or proprietary client-server data channels. The hosts' data is collected, and the tool performs diagnostics.

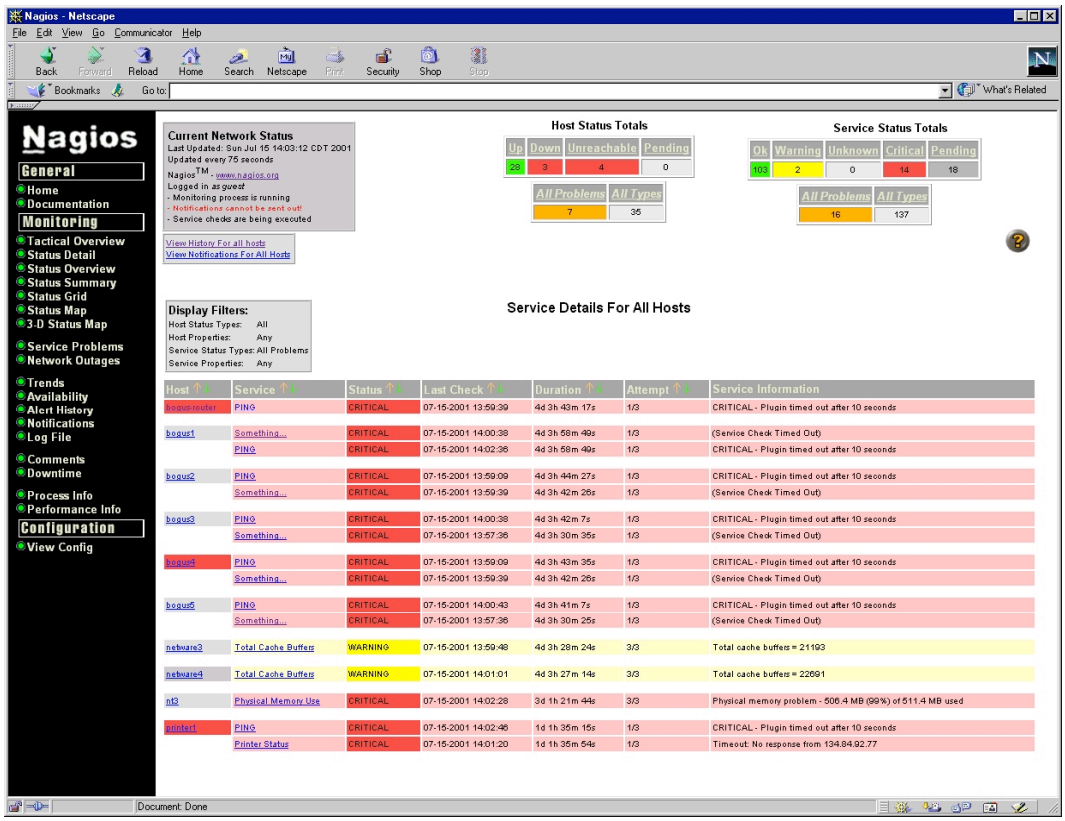


Figure 4: Nagios's service status view. Nagios reports host conditions primarily via text with color and flashing to emphasize critical items. Thus I classify it as an internal host dashboard (IH.DA).

At a minimum, these tools will check to see whether hosts are operating. Additionally, they may tell what processes are running on each host and compare this to what is expected or allowable. They may also check boundaries on any other condition that can be sensed at the individual hosts such as motherboard temperature.

As an example of the preattentive features of a dashboard, Big Brother, which shows all the monitored hosts on a single web page, normally displays a green backdrop, but if any host develops a serious problem, the backdrop will go red. Similarly, each host name is assigned an "indicator light" that goes red or yellow if there are any problems on that host. Big Brother, Nagios, and HP OpenView support drill-down to the process or alert level.

3.2.1.3 Summary Charts and Graphs (IH.CG)

Chart and graph internal host views were mentioned by only two of the 20 interviewed users. Tools in this category include network management tools such as eHealth (Figure 5) and performance

analysis tools like Orcallator. Although these tools may present much data in text form, their primary usage involves their graphing capabilities.

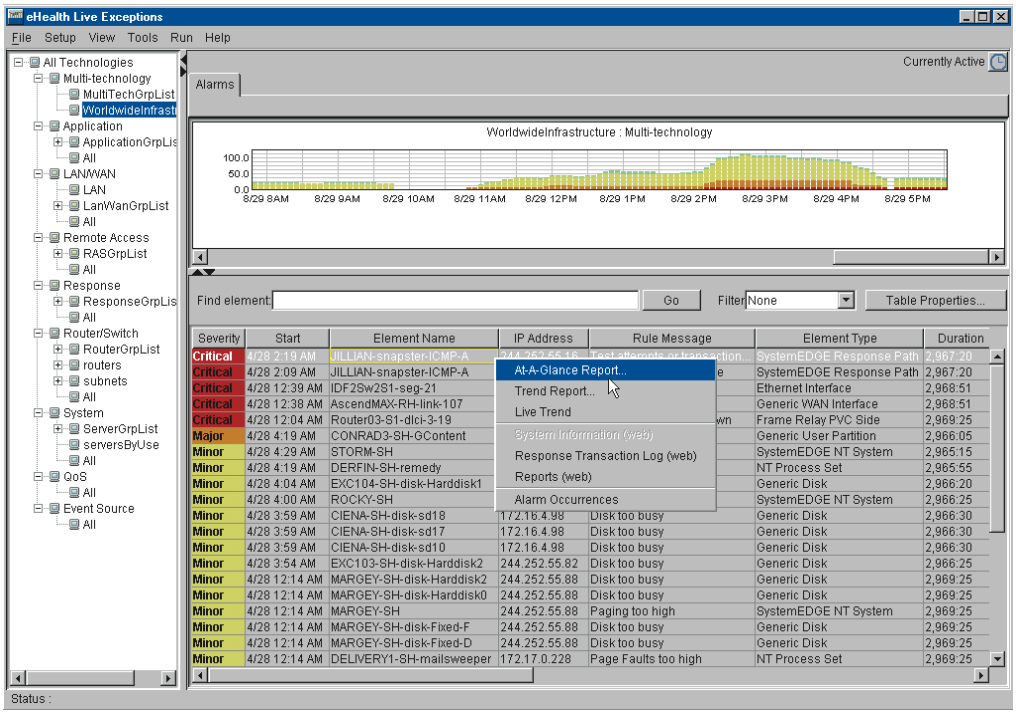


Figure 5: eHealth's Live Health Browser[3].

The eHealth suite of network management tools from Computer Associates[30], provides a multiple internal host view of medium and large networks of hosts using SNMP traps and queries. The product uses changes in its source data coupled with architectural configuration details of the network (provided by the user) to perform "root cause analysis" when traffic congestion problems occur. This analysis provides an overview of the management state of the system. The eHealth tools do not monitor traffic flows and relate them to host activities; rather they analyze host performance data and then use artificial intelligence to infer causes of potential network outages. Thus EHealth tools are essentially loosely correlated internal host views rather than network or end-to-end views.

Orcallator[31] is a single internal host view that produces charts and graphs for performance analysis. Orcallator displays data from the Orca SE Performance toolkit that includes all kinds of host-based quantities, including network interface performance. Although Orcallator plots network interface performance statistics, it treats the interface as just another device (tallying the throughput, number of connections, etc.) and is not aware of the identities of the hosts on the far end of the connections. Thus it is merely an internal host view, not a networked host view.

3.2.1.4 Visualizations (IH.VZ)

Very few visualizations of the internal workings of computers exist, and still fewer are applicable to computer security awareness. None of the interviewees used tools of this category. I was able to identify only one tool, the Visible Computer Applet[32], that fits this classification, and it is more education-oriented than computer security-oriented.

[3] Copyright © CA. All rights reserved. Reprinted by permission.

3.2.2 Networked Host Views (NH)

Networked host view tools were also mentioned by 14 of the 20 interviewees. Unlike the internal host view, the networked host view is very applicable to visualizing communications for security because these tools are aware of individual network connections and activity. They do not gather any network traffic, but several prominent ones do correlate process and network activity (from the perspective of the local host only), a key component in this research.

3.2.2.1 Text-Based (NH.TB)

All the networked host view tools mentioned by interviewees were text-based. Two such command-line tools provide textual snapshots that can relate the activities of local processes to network activity. These tools, lsof and netstat, are indispensable helps for users to create a mental model of the end-to-end view although they do not provide any end-to-end representation themselves.

The lsof[24] utility, created by Vic Abell of Purdue University, lists all open files related to processes running on Unix machines. Because communication mechanisms such as pipes and sockets are considered files on Unix, lsof can provide a very effective, if difficult to understand host view of communications. For instance, typing "lsof -i -U" lists all the processes with open Internet sockets and Unix pipes. Lsof can report the process IDs, the file control block information, command names, and many other pieces of information that an expert can piece together into a very complete view of host communication activities. One important limitation of lsof is that it works by polling rather than by continuously watching the file system. Thus, lsof may not show connections that are created and destroyed between polling intervals. Although polling intervals can be shortened to a single second, data collection takes a relatively long time and may block at certain system calls.

The netstat utility first appeared in BSD UNIX version 4.2 and has subsequently been added to DOS, Windows, and other operating systems. The netstat command textually displays the contents of various network-related data structures. There are a number of output formats of the command, including: a list of active sockets for each protocol, protocol traffic statistics, remote endpoints, and routing information. While lsof can display what files are open due to network activity, netstat can show the contents the data structures available only in the kernel. Like lsof, netstat is also built on the polling model.

One noteworthy exception to the polling model is DTrace[33], a programmable infrastructure implemented by Sun for its Solaris 10 operating system to allow dynamic instrumentation of kernel activity. DTrace was brought to my attention only very recently, but it is so revolutionary it deserves inclusion. DTrace is much more than a networked host view; it allows visibility and instrumentation of kernel and library code on the fly. Since its initial publication[33], DTrace has undergone many improvements. In recent correspondence with DTrace's chief architect, Bryan Cantrill, I learned that one could write a DTrace script to correlate packets to processes and produce text output similar to what HoNe's infrastructure generates. While I learned of DTrace too late to build my visualization on top of it, I believe it could form an excellent basis for future development. DTrace is being ported to other Unix-like operating systems, so leveraging it could provide a means for HoNe's infrastructure to become cross-platform.

3.2.2.2 Dashboard (NH.DA)

Relatively few types of dashboard presentations appear in current use, and none of the interviewees mentioned using any. A recent tool in this category is NVisionCC[11]. NVisionCC is intended to be a compact networked host view dashboard for showing process status on large cluster computers. It focuses on the state of the processes monitored, rather than their communications activities. Currently, NVisionCC includes three elements: a Process Monitor Module that tracks the processes running on

each node, a Port Scanner Module that scans each node for open network ports, and a File Integrity Module that validates the identity of disk files, particularly those thought to be targeted by hackers. To become a true end-to-end view, NVisionCC would need to be tightly integrated with a related tool, NVisionIP[34], that provides the network view. Even with integration, unless NVisionCC linked the process activity on the monitored hosts to network activity, the result would still not provide a true end-to-end view. Although NVisionCC's main view has many characteristics of a true visualization, I have categorized it as a dashboard because the bulk of the information presented is in text form. Depending on how the implementation actually turns out, it may be better classified as a visualization.

3.2.2.3 Summary Charts and Graphs (NH.CG)

Very few summary chart and graph presentations of the networked host view exist, and again, none of the interviewees mentioned any tools of this sort. I found one tool, gkrellm[35], that is as close as I could find to a member of this class. Gkrellm is a modular monitoring program with a plug-in architecture that can display graphs for any desired quantity. The built-in monitors include a CPU utilization monitor, a process-forking monitor, and a TCP port activity monitor. Together, these three monitors can provide a rough, single-host perspective of a host's communications activities. Gkrellm provides no linkage between communications activities seen on the network interfaces and the activities of processes in the monitored system. It is arguable whether gkrellm belongs in this category or in the internal host view category. However, with the proper plug-ins, gkrellm could be a good fit here.

3.2.2.4 Visualizations (NH.VZ)

True visualizations of the networked host view have been implemented, but none exist that are useful for computer security awareness or incident response. One outstanding visualization, Rivet: The Visible Computer[36] (see Figure 6) does provide a true visualization of every aspect of one or more computers. The Visible Computer is actually an application of Rivet, which is a general-purpose visualization framework. The visualization presents live data from SGI's Performance Co-Pilot (PCP)[37].

Figure 6: Rivet: The Visible Computer.

Rivet's Visible Computer provides quite as much detail as could ever be needed to analyze the security implications of any activity a host sees from the net—in fact, it presents too much information. System administrators do not have time to conduct indepth analyses; they must locate and fix problems quickly. Unfortunately, Rivet no longer exists as a networked host view; it has been subsumed into the Polaris database visualization suite[38].

3.2.3 Network Views (NW)

Most administrators I spoke with considered utilities that view network traffic fundamental, and 13 of the 20 interviewees mentioned using one of these tools. So important is this network view to network security monitoring that Richard Bejtlich[39] writes, "A product is worthless unless it can see packets." Network views gather and present network traffic (e.g., packet headers) to provide insight into network events. They may provide some level of "intelligent" analysis about the traffic, but they do not look inside hosts on the network.

3.2.3.1 Text-Based (NW.TB)

All 13 interviewees who mentioned a network view tool mentioned at least one text-based tool. Two such utilities, tcpdump[40] and Ethereal[41] (Figure 7) are basically promiscuous packet sniffers, collecting any communications traffic on the network segment where the listening host resides.

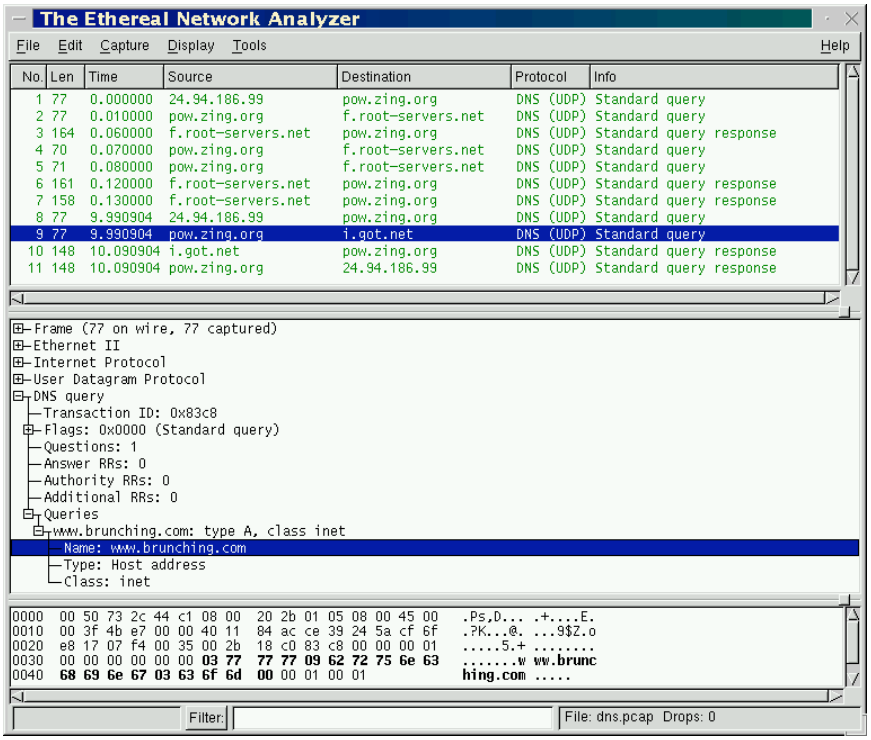


Figure 7: Ethereal's Main Window.

These tools present a textual dump of the packet headers and contents to the user. The user can determine what machines are talking on the network and to whom they are talking, providing a network view. Although Ethereal has a graphical user interface, the information it presents is primarily textual. Ethereal does provide some charts and graphs, but the capability is so primitive compared to other graphing tools that I retain Ethereal in the text-based category.

3.2.3.2 Dashboards (NW.DA)

No interviewees mentioned using a dashboard network view, but in recent months (since the interviews) a very good tool from this category, Sguil[42] (Figure 8), has begun to gain ground. Sguil (pronounced "sgweel") is a network security analysis tool built by network security analysts. The main component is a dashboard for displaying real-time events from Snort IDS[43]. Sguil includes other components that facilitate the practice of Network Security Monitoring (as defined in [39]).

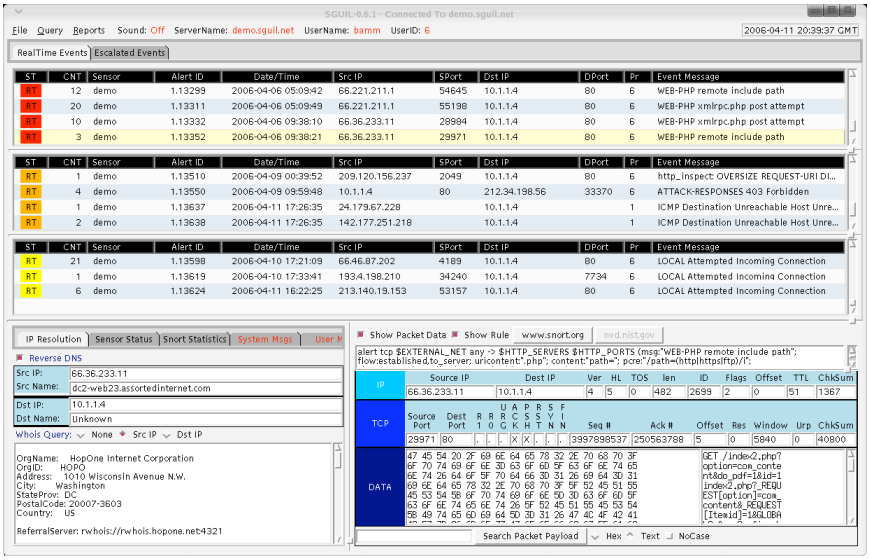


Figure 8: Sguil's Real-Time Events window.

Sguil has a client-server architecture; thus a Sguil client may connect to one or more Sguil servers and vice-versa, allowing many analysts to work on the same data in tandem. Sguil is among the first user-directed network view tool that allows users to control the amount of traffic context that they use to determine the security implications of an event.

3.2.3.3 Summary Charts and Graphs (NW.CG)

A popular solution in this category is the Multi-Router Traffic Grapher (MRTG)[44]. Although MRTG was originally specifically a network view solution, it has become a general-purpose graphing tool that no longer fits cleanly within any category of this taxonomy. Other tools use MRTG and similar software to present a graphic view of the monitored network. One such tool is AutoFocus[45]. Estan, Savage, and Varghese have designed the AutoFocus tool to reduce the complexity of traffic reports by aggregating all the IP address entries into CIDR blocks according to the amount of traffic they generate. AutoFocus shows only the "heavy hitters"—aggregations of addresses whose total traffic exceeds a certain threshold. AutoFocus does not integrate any data from within hosts on the network, nor does it provide any visualization beyond MRTG-style charts and graphs. One could argue against AutoFocus being classified as NW.CG because it relies heavily upon tables of text, but I have left it here because of the analytical value of its charts. Its major contribution is a novel traffic aggregation algorithm that greatly reduces the size of the traffic reports administrators must read.

Another chart and graph network view is S-Net[46]. S-Net provides a large variety of statistical plots of packet traces that can give useful insight to system administrators about the state of their networks. But the statistical approach provides a highly abstracted and condensed view of the network separating quantities from their source data and making drill-down to the root cause difficult.

3.2.3.4 Visualizations (NW.VZ)

While only one interviewee mentioned using a network view visualization, many appear in the literature. There are two primary kinds: packet-header visualizations and visual Network IDS (NIDS). The typical packet-header visualization displays source and destination IP addresses, source and destination TCP/UDP ports, and protocols using a two or three-dimensional scatter plot. NCSA visualizations NVisionIP[34] (Figure 9) and its cousins[47][11] fit into this category as does [48]. NVisionIP is a high-density, pixel-map visualization of net flow data for an entire Class-B address space (some 65,535 hosts).

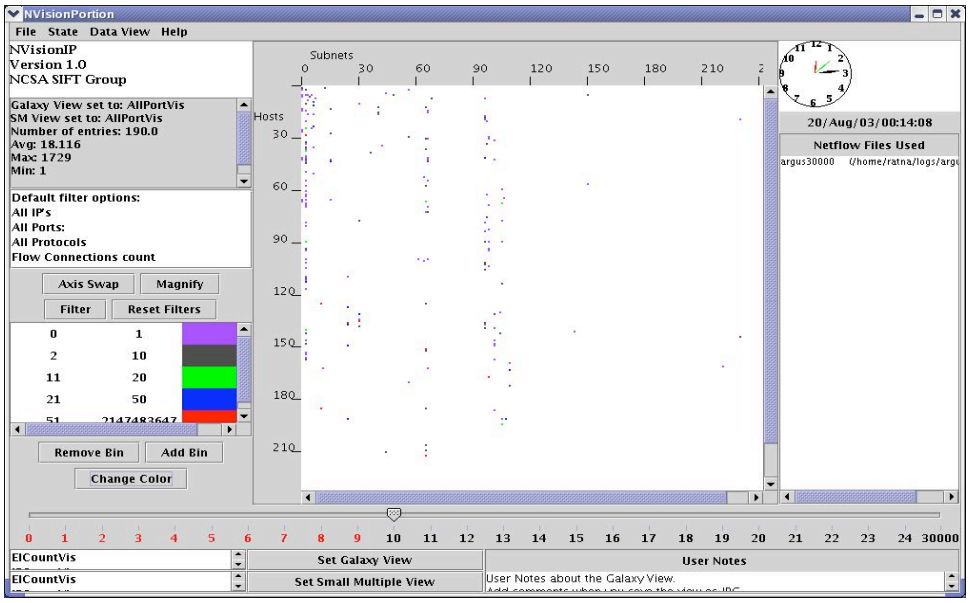


Figure 9: NVisionIP's "Galaxy View."

As shown in the taxonomy, a visual NIDS plots alerts from various network sensors on a framework indicating the location, function, or criticality of the monitored system. Examples of visual NIDSs are the Spinning Cube of Potential Doom[49], and Secure Decision's Secure Scope[50] (Figure 10). NVisionIP's "galaxy view" and VisFlowConnect[47] are both network views, relying solely on net flow data without any host process data at all.

3.2.4 End-to-End Views (EE)

3.2.4.1 Text-Based (EE.TB)

As mentioned in section 3.2.2.1, DTrace could be used as a networked host view. Thus it is conceivable that two or more hosts running a DTrace networked host view script could combine their output into a text-based end-to-end view. Of course, neither the networked host view scripts nor the infrastructure to join them exists, but the capability is worth mentioning.

3.2.4.2 Visualizations (EE.VZ)

The Host-Network Visualizer provides the only end-to-end visualization of communications data. When viewing data from a single monitored host, HoNe is more properly a Networked Host Visualization (NH.VZ). But with multiple monitored hosts, it becomes possible to see the processes on one host communicating across the network to processes on another monitored machine. Thus we may see the entire scope of the communication rather than just a network or host view alone.

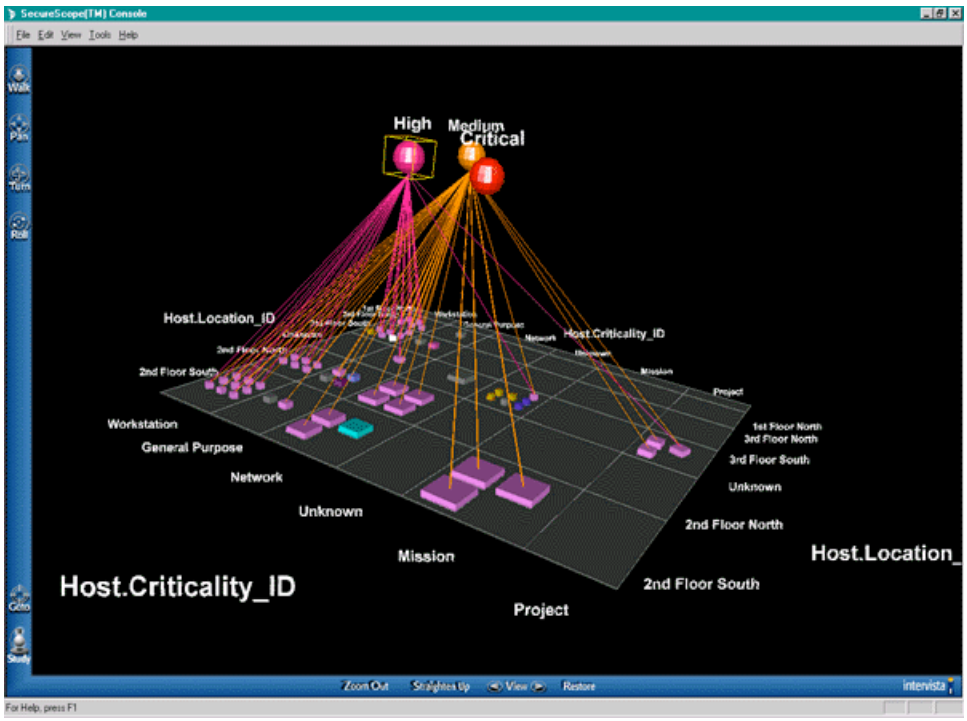


Figure 10: Secure Decisions' Secure Scope is a visual network intrusion detection system.

Conclusion

In Chapter 2 I demonstrated that bridging the host/network divide was important to my users and that there were two aspects to the bridge: technical (creating new data paths within the kernel), and cognitive (allowing people to see and understand the linkage between host processes and the network). Loosely speaking, the technical bridging corresponds to the communications context while the cognitive bridging corresponds to the presentation mode presented in my taxonomy. In this chapter I have shown how no other tool bridges both dimensions of the divide as does HoNe with its infrastructure and its visualization. HoNe is the only true networked host view that provides a practical visualization for computer security applications. Combining multiple HoNe networked host views together forms the first and only end-to-end visualization of TCP/IP communication data.

CHAPTER 4 THE ROAD TO SOLUTION

Following the principles of participatory design[10], I gleaned wishlists, anecdotes, wisdom, and ideas from my interview subjects and then immediately turned these communications into paper prototypes and asked the subjects to evaluate them. Thus I involved the subjects as user/designers in the development project. I derived my prototypes directly from designs produced in collaboration with the interviewees. Every feature of the overall design and the prototypes was derived from interactions with interviewees and application of HCI methods and information visualization techniques. My prototypes also stimulated further design discussions with my user community.

4.1 Low-Fidelity Prototypes: Network Eye

Understanding my user community's needs required an ongoing dialogue involving further interviews and reviews of prototypes. Early on, my impression was that the users needed a high-density visualization of their networks that could show tens of thousands of hosts simultaneously grouped by degree of trust. I called this set of concepts, "Network Eye." I built and presented to them a number of low-fidelity, PowerPoint prototypes to reflect back to them what I though their input was telling me. I also made a few vertical prototypes to investigate the utility of certain concepts more precisely. This section briefly presents several of these prototypes and shows how the concept matured beyond the "Network Eye" phase.

4.2 Network Pixel Map

My objective was to display tens of thousands of IP addresses using pixel-oriented visualization techniques[51]. Each pixel of the map would stand for an individual machine seen on the network. I experimented with different layouts of the trust hierarchy and presented them to the users. Originally, I had hoped to fit a million machine icons on a single 1000x1000 pixel display, but as my understanding of the users' needs and how they would want to interact with my display evolved, I eventually scaled down to showing about ten thousand machines at a time. Users occasionally wanted to use the display to get information about individual hosts, and single pixels were too hard to focus on[52]. The result was the Network Pixel Map illustrated in Figure 11 and Figure 12.

4.2.1 Background

4.2.1.1 Assumptions

First, I assumed the existence of a known set of hosts that can be thought of as the "home" set; that is, machines that the user believes are benign and well-managed. I assumed nothing about the size of the home set except that there are probably fewer hosts in the home set than outside it. Although the majority of traffic observed on a given network is local-only, over time the majority of the IP addresses seen will likely be from outside the home set.

Second, I assumed data was collected within the home set and would be biased toward it. Network data is gathered using packet "sniffers," hosts that collect traffic on the network. Especially in wired networks, the traffic visible to a single sniffer is highly localized. Data on other network segments cannot be sniffed without port mirroring or other techniques. For this stage of my study, I assumed that only a few sniffers would provide the data to visualize and that all the sniffers would be located within the home set. This would bias the data collected by limiting it to traffic originating on the home network or destined for it (traffic passing through it would be routed around the sniffers by the border router). Thus I expected not to see traffic from one external host to another, but I did expect to see external to internal and internal-only traffic. These kinds of traffic patterns fit the polar layout more naturally than the Cartesian.

Third, I assumed communication seen within the home set is mostly internal. I used a polar layout with the home set near the origin to help highlight Internal←→External and Internal←→Internal traffic flows.

Finally, I assumed that network security analysts prefer to examine network data as close to real-time as possible. As a result, the presentation of data for my application must not rely on time-consuming preprocessing or computationally expensive space-optimized layout. Instead, I used the known characteristics of a host (e.g., IP address, trust category, etc.) to place it directly on the plot. In case of collision with an already plotted host marker, I moved the new marker to a nearby empty space. Once a marker was plotted, it stayed where it was placed until traffic to and from it disappeared from the network.

4.2.1.2 Layout

Marker (host) arrangement is critical to user insight in high-density displays as noted in [51]. Each marker represents a unique IP address observed in the network traffic. Marker placement indicates the trust level and IP address of the host. Each IP version 4 address is thirty-two bits long and is written as w.x.y.z where w, x, y, and z are eight-bit unsigned integers called octets. My first attempt was to plot each IP address in Cartesian coordinates using the first two octets of the address as the abscissa and the other two as the ordinate. While this method allowed users to relatively easily locate the general position of a given IP address on the plot, it did not lend itself to the idea that some of the hosts are locally administered, home hosts. Home was wherever your IP address mapped to on the plot.

I decided to use a polar layout of these markers to show the hierarchical nature of trust relationships where every address displayed in an inner group is trusted more than the most trusted member of any group outside it. Using polar coordinates, it was more natural to locate home in the center of the plot. In interviews with my user community (see section 2.1) and a usability study of a prototype security awareness tool[53] I learned that system administrators see the world of machines as falling into two basic trust categories, "us" and "them." These trust categories may be broken down into any number of arbitrary categories on a continuum from most to least trusted. With polar coordinates, a natural representation of trust levels would be nested circles with the most trusted level at the center.

Figure 11 shows the polar trust layout that I implemented. I planned the Network Pixel Map to accommodate up to five trust levels, here arbitrarily called: Home, Trusted, Safe, Unknown, and Danger. Another way to think of trust is organizational proximity. I arranged these levels, target-style, with the most trusted hosts' markers (Home) placed in the center ring and the rest of the markers placed further from the center as trust level decreases. Trust levels may be assigned from network configuration information and via experience. Dangerous hosts would be assigned from local IP blacklists or moved to the outskirts as they are discovered. Figure 12 shows how the Network Pixel Map would look showing about 1,000 real IP addresses.

The Network Pixel Map could be used in a number of ways as a security-monitoring tool. I could assign different meanings to the hue, saturation, and brightness of the markers to indicate how recently the machine was seen, how high the risk of communication with that host seemed to be, or how much bandwidth was involved in recent communications with the host, for example.

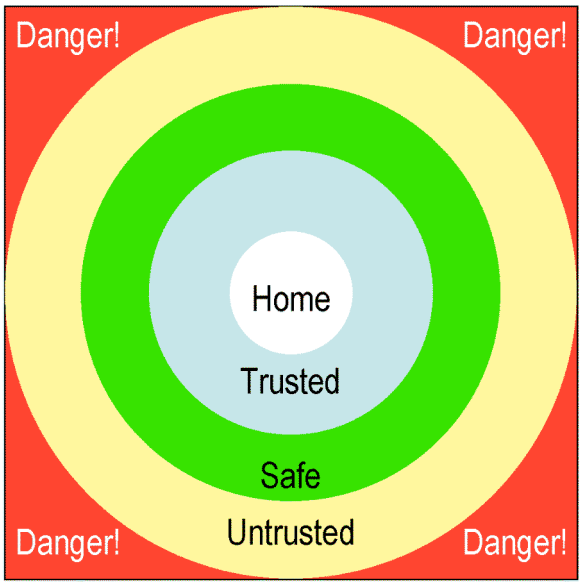


Figure 11: Network Pixel Map trust layout

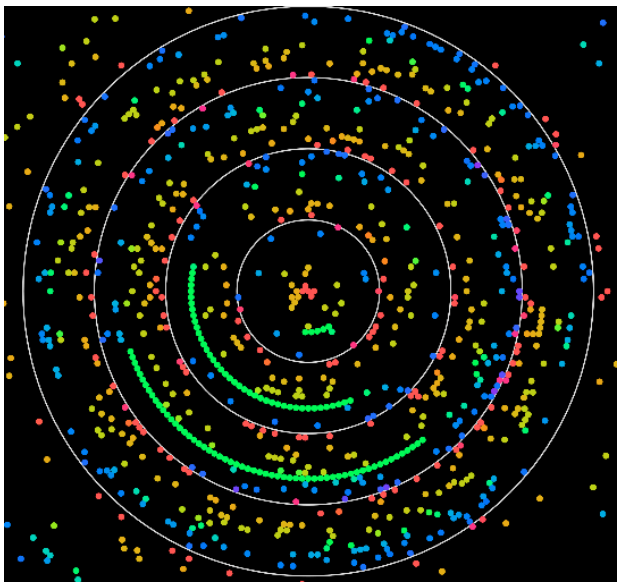


Figure 12: An example Network Pixel Map

4.2.1.3 Root Polar Coordinates

Unfortunately, I encountered two major problems with normal polar coordinates: distortion and severe occlusion. However, polar coordinate plots have several serious problems including distortion, occlusion, and reduced area. I used a novel adaptation of polar coordinates where I plot the square roots of ρ and θ rather than the raw values as in normal polar plots to alleviate all three of these problems. Figure 13 shows how I map an IP address using Cartesian, polar, and root polar coordinates. I will briefly treat this subject here, but for more details and the mathematical underpinnings of the method, please refer to my paper on root polar coordinates[54].

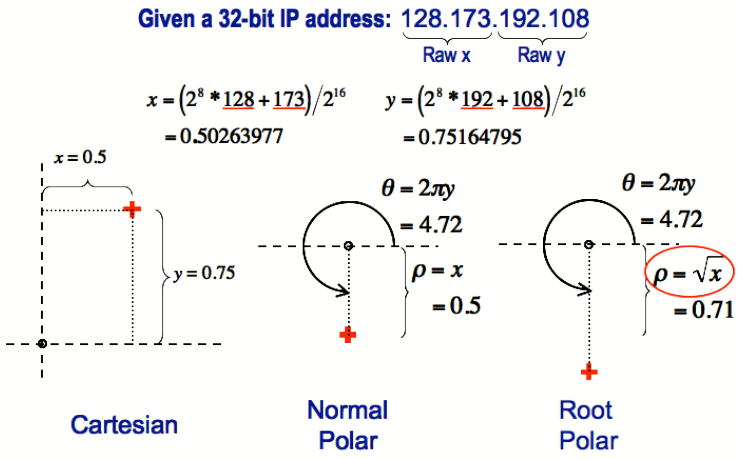


Figure 13: Derivation of Cartesian, polar, and root polar coordinates from an IP address.

Distortion in polar plots.

To alleviate the distortion induced by polar layout, I tried exponential and logarithmic smoothing, but these functions only made the problems worse. What I needed was a function of ρ that grew more

rapidly than linear at first and smoothly slowed its growth rate to the edge of the plot (at ρ =1). A simple function that behaves this way is the square root function (see Figure 14). The early rapid growth of sqrt(ρ) causes the points near the center to be spread out more. The later slow growth (relative to linear) causes the point density to increase near the periphery. This shape naturally counteracts the density distortion that arises from plotting in polar coordinates.

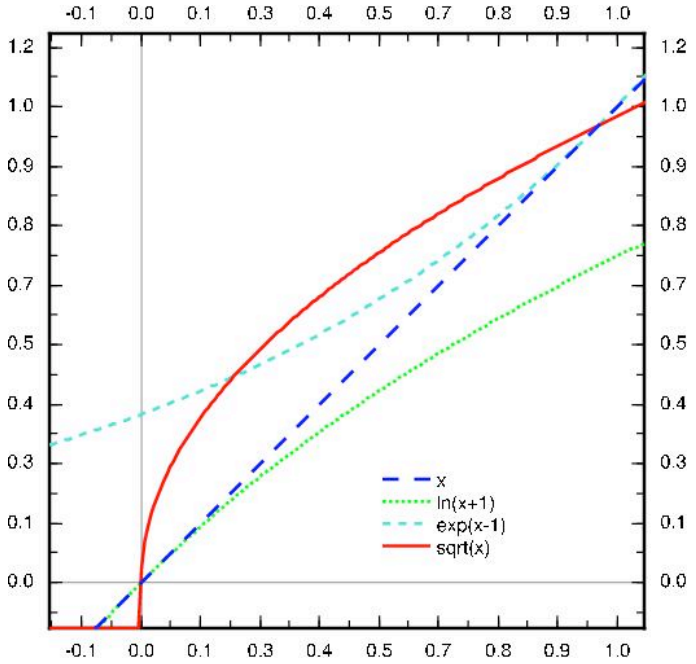


Figure 14: Graph of *x*, *ln*(*x*+1), *exp*(*x*-1), *sqrt*(*x*), and *x* in the interval [0,1].

Since I could constrain ρ to be between zero and one, I did not have to worry about taking square roots of negative numbers. The resulting "root polar" coordinates worked very well both at removing the artificial clustering at the center and smoothing out the variance of the distribution of markers throughout the plot. Root polar plots are also not greatly affected by the plotting order, removing the reliance on a priori knowledge of the data or pre-sorting.

Normal polar plots distort the data by compacting it near the center and spreading it out at the edges. This characteristic has been useful in certain applications such as retinal emulation for robotic vision[55], but for my application the distortion destroyed the picture I was trying to show. I wished to plot the home set of IP addresses at the center of the plot, but I also wanted to accentuate the individual members of the home set. Although the home set was much smaller than the set of external addresses, I wanted to give the home set equal space.

The distortion due to polar coordinates can be quantified as the variance of the plotting density, *var(d)*. Uniformly spaced (n points at intervals of $(2^{32}-1)/(n-1)$) unsigned integer data plotted in a rectilinear plot has a constant plotting density. That is, *var(d)* approaches zero as the regularity of the grid approaches perfect uniformity. However, scaling the data to fit it on a 1000x1000 pixel screen introduces round-off error that will never allow us to attain perfect uniformity.

To estimate *var(d)*, I took a number of samples by placing a grid of constant-sized, nonoverlapping tiles over the whole plot area (see Figure 15). I then counted the number of markers that overlapped each tile and estimated the population variance of the number of markers found over all the samples.

In contrast to the near uniformity of a Cartesian plot of uniformly spaced data, a polar plot is visibly denser near the center and sparser near the periphery. *Var(d)* is two to three orders of magnitude larger for polar plots than for Cartesian. However, I was surprised to find that *var(d)* for root polar plots was only one order of magnitude greater. To refine my observations, I conducted a study using the network pixel map displaying the same set of uniformly spaced IP addresses on Cartesian, polar, and root polar coordinates. I compared several plots of the same data, measuring the density variance of each graph. I found *var(d)* for the normal polar plot was consistently higher, by an order of magnitude or more, than the same data plotted on either Cartesian or root polar plots with the same area.

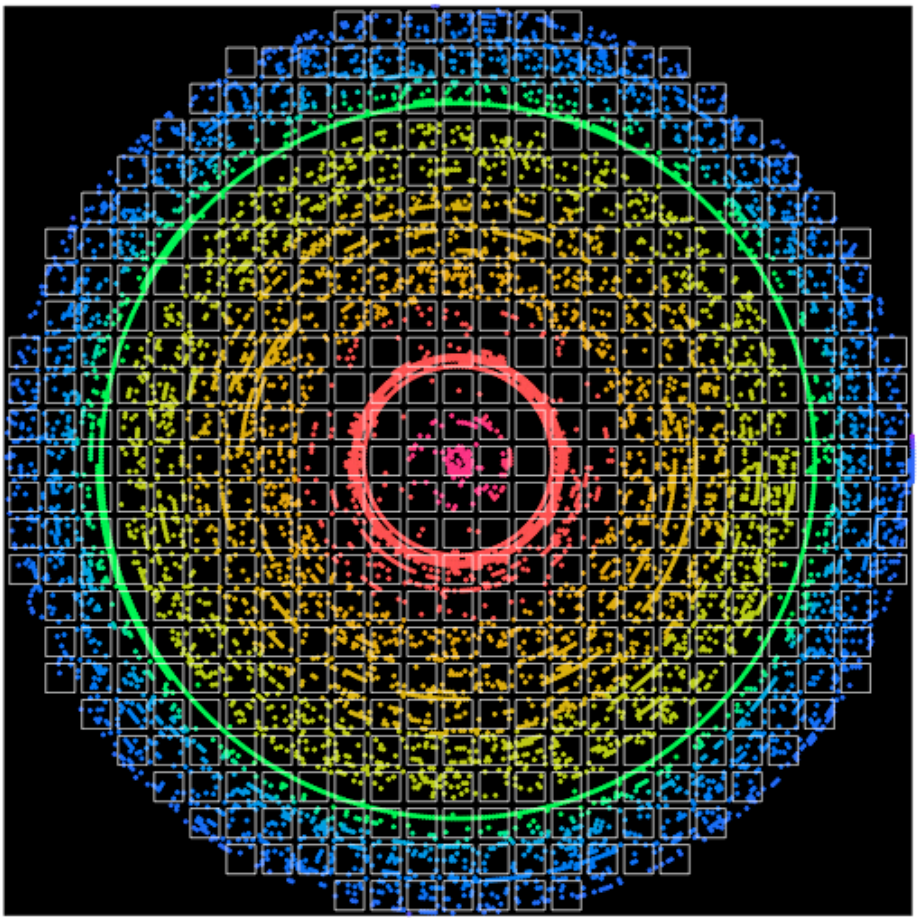


Figure 15: An example of how I sampled density on the completed plots. Each white rectangle represents a density sample. I counted the number of markers overlapping each rectangle and computed the variance. I used this metric to compare density distortion between plot types.

After numerous density experiments, a pattern appeared to me: Root polar coordinates seemed to pack the data nearly as uniformly as Cartesian plots could. But root polar coordinates had the advantage over Cartesian coordinates of being able to naturally plot trust bands in concentric circles. As Figure 16 shows, root polar coordinates inherit the advantages of both Cartesian and normal polar plots.

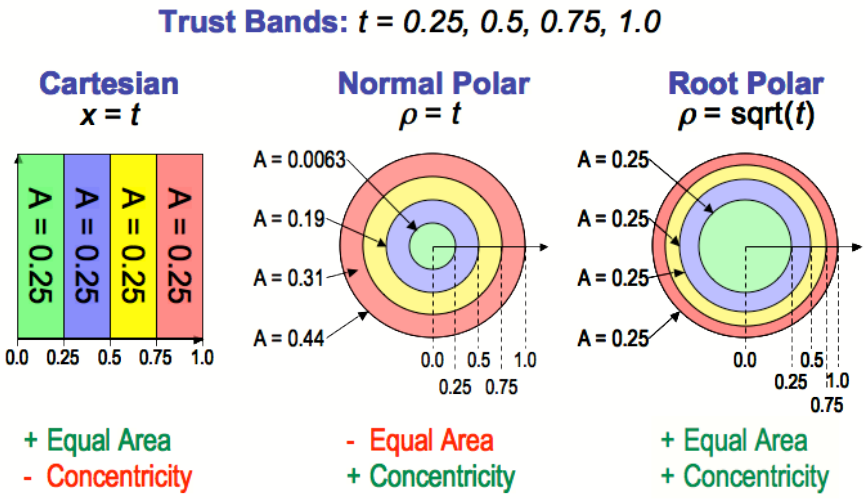


Figure 16: Comparison of trust band areas for Cartesian, polar, and root polar plots.

Occlusion in polar plots.

The central clustering of polar coordinates caused a second problem: occlusion. With large numbers of points, a polar plot may have many layers of overlapping markers at the center, most of them hidden by more recently plotted markers. Occlusion is a general problem that happens in all kinds of graphs, but the problem is exacerbated in polar coordinates near the origin. I can fix occlusion by detecting collisions and moving the markers around, but this makes the perceived density near the center of the polar plot appear worse (this method was used in the normal polar plots in Figure 17). With collision resolution, the "solid" central area looks twice the diameter of the same plot without collision resolution, revealing how much overplotting is happening in a normal polar plot.

As it turned out, root polar plotting alleviated much of the problem with occlusion found in normal polar plots. Over-plotting still occurs, though, whenever there are potentially more markers than unique slots on the graph. With a plotting domain of 2^{32} IPv4 addresses, no known display is large enough to completely avoid occlusion. However, I employed a nearest neighbor algorithm to resolve collisions when they occur and prevent occlusion until the plot fills to capacity.

Figure 17 shows a series of plots of identical uniformly spaced IP address data. I colored the markers using a Keim's HSI rainbow gradient[51] with lower numbers at the red end of the spectrum and the higher numbers toward the violet end. Plots on the right side of Figure 17 allowed points to overlap. Those on the left side of Figure 17 used collision resolution to fix any overlap. The first two plots are Cartesian coordinates, the second two are normal polar coordinates, and the third two are root polar plots. Note the visible differences in *var(d)* between the graph types.

High *var(d)* indicates greater distortion in the plotting density. Collision resolution increased this distortion in Cartesian plots slightly, but the difference was mostly due to the plotting round-off error that my nearest-neighbor algorithm corrected. Collision resolution in normal polar plots dramatically reduces the plotting distortion in normal polar plots because there are no longer areas where high overlap is allowed. But the result is a large solid area near the center of the graph while the edges are still relatively sparse. Normal polar plots with collision resolution still have large density distortion at least one or two orders of magnitude greater than Cartesian plots of identical data. But with root polar coordinates, the difference in density distortion due to collision resolution is very slight. Theoretically,

the density distortion is the same as that of Cartesian coordinates, although in practice I have found it to be within one order of magnitude or less.

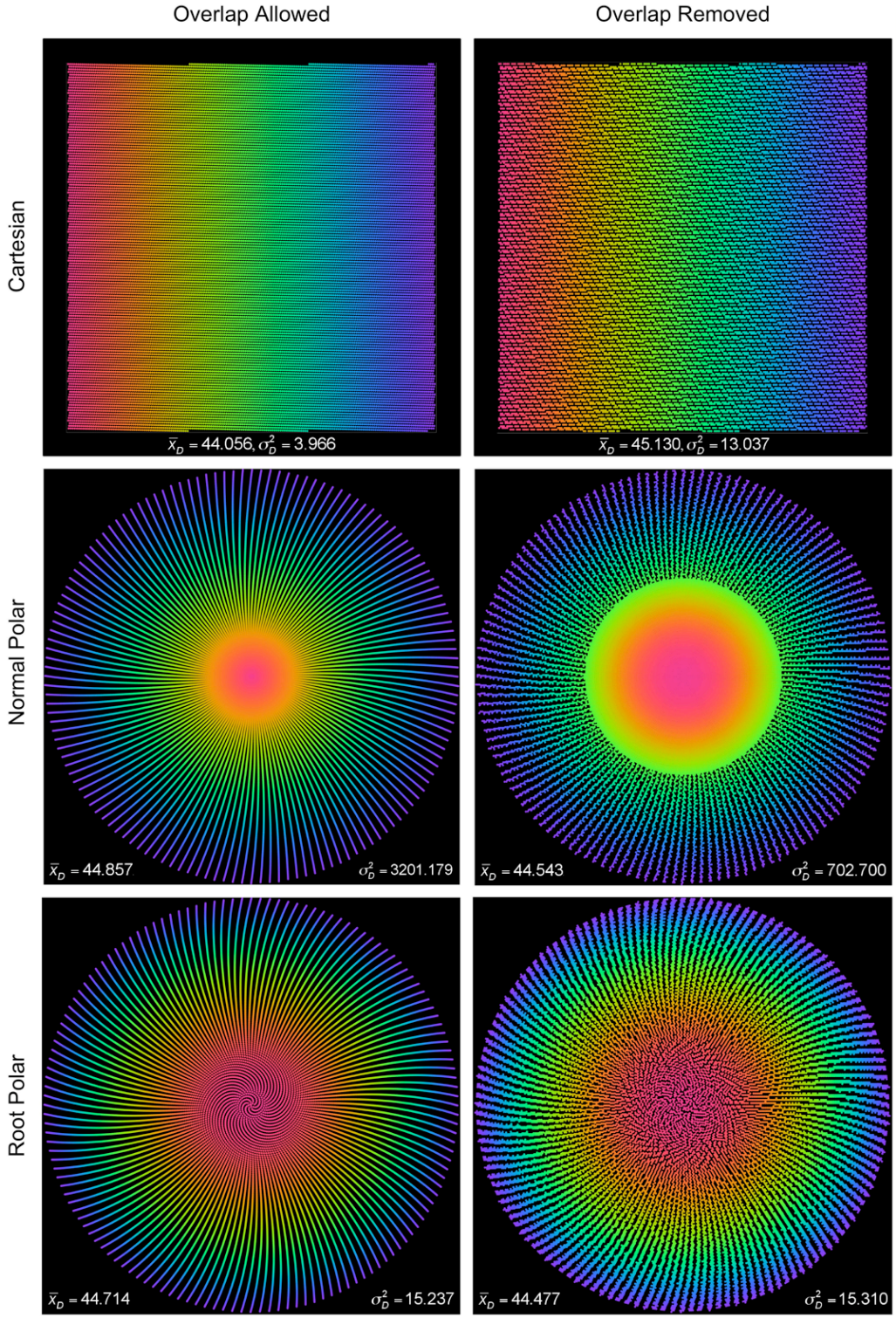


Figure 17: A series of plots of identical uniformly spaced IP address data showing the visible differences in *var(d)* between the graph types with and without collision resolution.

Smaller area of polar plots.

One intrinsic problem with polar coordinates plots is that they have only π/4 of the area of a square Cartesian plot whose sides are the length of the polar plot's diameter. I can mitigate the effect of the smaller plot space for my application by placing some markers (for the least trusted hosts) in the otherwise empty corner areas. Not all applications can make sensible use of the corner spaces, so they are either constrained to a smaller area or are subject to being clipped. In my study, to make the comparison between polar and Cartesian coordinates more clear, I elected to compare polar plots to Cartesian plots of identical area.

Putting Home in the Center

The real advantage of root polar over normal polar plots was clearest in the problem of keeping the "home" network at the center of the plot. I placed markers in trust level "bins" while keeping the machines the user is most concerned with (his own) in the center (please refer to section 4.2.1.2 for an introduction to trust levels). Not all hosts on the Internet are trusted equally. In fact, most organizations have a "white list" of address spaces they administer and a "black list" of address ranges known for previous malicious behavior. Between the two is the murky set of addresses that are simply unknown. Of the unknown addresses, there are probably some that an administrator may expect to be commonly accessed by his users, such as search engines. An administrator within the organization may also subdivide the organization's "whitelist" into a set of machines he administers (and thus has a personal stake in) and another set with the rest of the enterprise's addresses. For my application, I define five trust levels. For the purposes of illustration I used: self, enterprise, safe, unknown, and dangerous. Of course an administrator may choose to use only the minimal set of trust levels: "us" and "them." The number of levels (as long as it stays reasonably small) is unimportant. Using trust levels, I plotted the most trusted ("home") nodes in the center to obtain plots like those shown in Figure 18. The plots show 8,513 real IP addresses collected from a single point on the Virginia Tech campus over a period of two weeks.

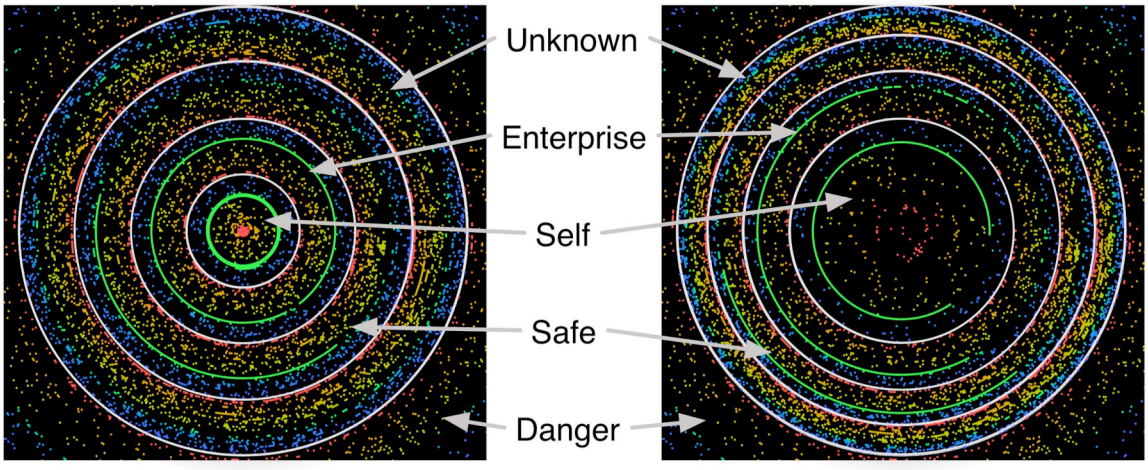


Figure 18: Normal and root polar plots of IP data with five arbitrary trust levels superimposed. The color gradient indicates the relative value of the addresses plotted (spectral, with lower values toward the red end and higher values toward the violet end). With trust levels, I plot addresses from low to high value within their trust band. All addresses within a given trust band should be (dis)trusted equally.

Root polar coordinates give the innermost circle the same area as the other trust bands, making differentiation of individual host machines easier. Each IP address has an associated trust level assigned

according to how its CIDR block is classified by administrative policy. An important difference between the plots may be seen in the green-colored markers in the innermost trust level. In the normal plot, the distortion coupled with collision resolution has forced these markers to circle about the origin several times. But in the root polar plot these hosts do not even complete a single revolution, even though there are many of them.

In fact, the reason the root polar method is so successful at dividing the plotting area into equal area trust bands is because it has a constant *packing factor*. Packing factor may be defined as the expected proportion of uniformly distributed points in a band divided by the plotting area of the band. With Cartesian coordinates, dividing the area into four trust bands gives 25% of the area to each, and one would expect that given a uniform distribution of points into trust bands, 25% of the points would fall into each trust band. Thus, the plotting factor for Cartesian coordinates is always 1. But this turns out not to be true for normal polar coordinates. As shown in Figure 16, above, when one divides the area into four trust bands, 25% of the points must be plotted within 0.63% of the plotting area for the most trusted hosts in the innermost band. The packing factor of the inner trust band is thus about 39.7. Conversely, only 25% of the points must be plotted in 44% of the area in the outermost band. This yields a packing factor of about 0.57. Thus we see that the packing factor for normal polar coordinates is not constant as it is for Cartesian coordinates. Figure 19 shows how the packing factor may be defined for normal and root polar coordinates and why root polar plots always divide the plotting area into equal area bands.

In Figure 19, the limits of an arbitrary trust band are t_0 and t1, where t0 < t1. For an extremely thin outer band, the packing factor approaches 0.5 for normal polar coordinates. Conversely, for an extreme inner band, the packing factor for normal polar coordinates approaches infinity. But for root polar coordinates, any trust band has a packing factor of 1, and given equally spaced trust bands, they will each have the same area. Another interesting point illustrated by Figure 19 is that the corner areas of the square enclosing the polar plotting area, taken together, equal about one fifth of the area of the square leaving four fifths of the space for the polar plot. Thus root polar coordinates can display five concentric trust bands of approximately equal area in a square display. This feature is handy when five trust levels are appropriate.

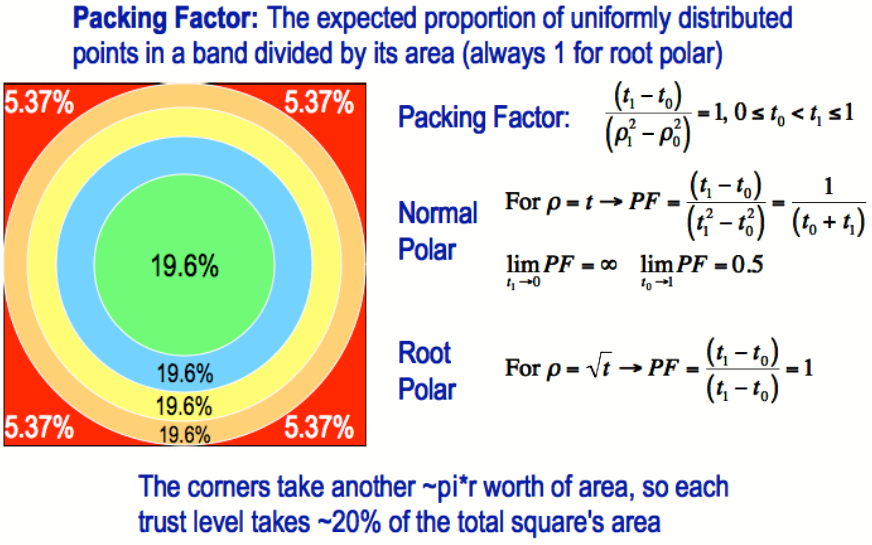


Figure 19: The definition of Packing Factor and how it explains root polar coordinates' success.

Another thing that could be done with root polar coordinates is to warp the relative areas of the trust bands by changing the power of the root used. The user can expand or contract the size of the central area by applying different powers of ρ instead of 0.5. Smaller powers of ρ expand the center and larger ones contract it. This usage is reminiscent of the "fisheye lens[56]" often used as a focus+context approach in information visualizations. However, I used root polar coordinates as a robust layout approach that minimizes occlusion over the entire graph rather than simply to magnify a transient focal area. Root polar plotting with trust levels provided a simple way to place the area of most interest to the user centrally while avoiding the distortion inherent in normal polar plots. My root polar coordinates work made the following contributions:

- *A new layout that can accommodate tens of thousands of hosts simultaneously*: The layout locates the set of home hosts in the center where users expect "home" to be, and it supports spatial grouping of multiple trust levels.
- *A way to overcome the plotting density distortion of normal polar coordinates*: Particularly, square root polar coordinates neither concentrate a large number of points near the origin, nor do they spread points out near the periphery. I believe this shows that root polar coordinates are most useful when occlusion would garble the message of the data. In my application, using root polar coordinates will allow the users to clearly see the "home" hosts without occlusion.
- *A means of plotting data in near real time without complex optimization*: Root polar plots help spread the data around naturally without having to resort to computationally expensive optimization methods. My study[54] showed that fewer collisions and thus less work to resolve them results when root polar plotting is used as opposed to normal polar plotting. I showed that root polar plots with collision resolution were quite robust when data is plotted in random order. Thus, root polar plotting is a good choice for presenting data in a polar layout under tight, near real-time constraints.
- *Empirical comparison of Cartesian vs. polar vs. root polar plots*: My paper[54] presented a numeric proof that the plotting density distortion of root polar coordinates was less than that of normal polar coordinates. I believe my conclusions will help visualization specialists to decide when to use normal polar, root polar, and Cartesian coordinates.

Key Conclusions: Discovering root polar coordinates showed that high-density visualization of large numbers of host on a single screen was possible. However, I learned that not all hosts were equally important to my users—the more trusted hosts needed to be separated from the lesser trusted ones because administrators cared the most about the hosts they operate. Using polar coordinates helped centralize the home hosts, but caused them to be jammed together into the smallest space. Root polar coordinates solved this problem by enabling me to divide a polar plot into an arbitrary number of equal-area, equally-dense, nested trust bands.

4.2.2 The Network Pixel Map Testbed

I developed a network pixel map testbed to investigate the usable limits of packing host markers onto a single screen (see Figure 20). This prototype was written in Tcl/Tk scripting language. The testbed helped demonstrate that plotting one million hosts on a screen was unusable and established the probable limit to be on the order of a few tens of thousands. The prototype also helped test the efficacy of polar versus Cartesian plots of host markers. The testbed helped me develop and test root-polar plotting and provided many insights on collision resolution algorithms.

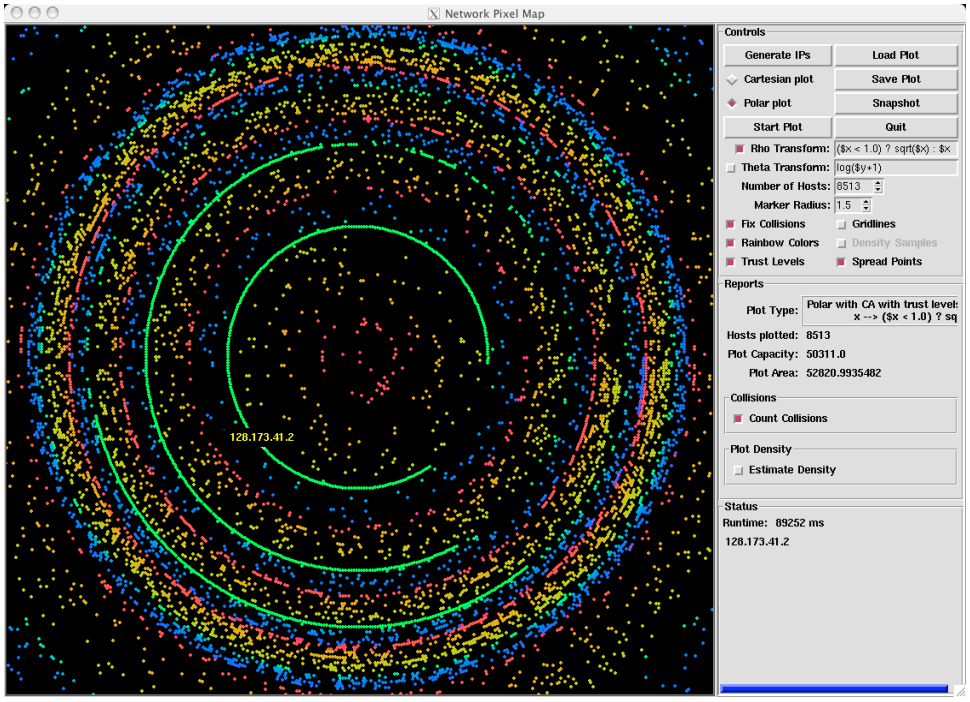


Figure 20: A screenshot of the Network Pixel Map testbed application

The testbed application allowed the user to make rectilinear or polar plots with arbitrary coordinate transforms. One can vary the number and size of the markers and either generate random sets of host addresses or import a list of hosts from an outside source. Plots may be made with or without trust levels and collision resolution was also an option. Once a plot is complete, the tool may make a record of the number of collisions, measure the mean and variance of the plotting density, and record the runtime. Gathering these variables, under a variety of conditions, I performed empirical experiments and a numeric proof of the assertions listed in the previous section.

However, the reaction of the user community to the network pixel map as a security-monitoring tool by itself was tepid. They were more interested in seeing what connections were being made between hosts. However, they were excited about the idea of joining two maps in tandem to monitor connections between hosts. This led to the Network Communication Diagram.

4.2.3 Network Communication Diagram and Alternative

The Network Communication Diagram showed connections across a monitored network from one or more perspectives. This visualization's high-density, three-dimensional scene was made of two mirrored network pixel maps connected by lines indicating connections between hosts. The Network Communication Diagram showed network traffic flowing between the TCP/IP client, on the left, to the server, on the right (see Figure 21). Elementary communication patterns such as point-to-point interconnections and fan-out/fan-in became readily apparent in this view. An alternative view that allowed users to see overlay networks that are communicating appears as Figure 22.

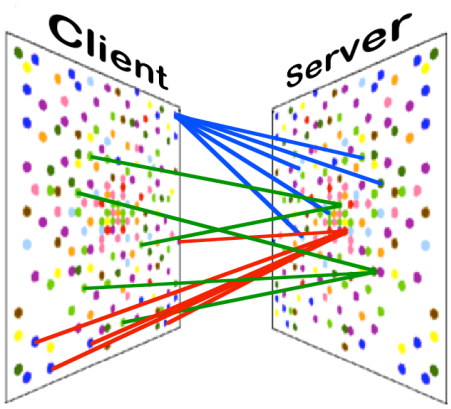


Figure 21: The Network Communication Diagram made fan-in and fan-out patterns easily visible.

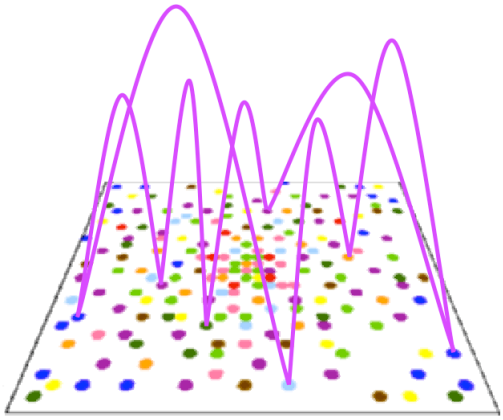


Figure 22: Alternative Network Communication Diagram that would show covert overlay networks.

Users were much more excited about the Network Communication Diagram than the Network Pixel Map. They were concerned that with any amount of traffic there would be too many lines and the view would be too occluded to use. This was especially of concern for the alternative view because the arcs would be even harder to follow than straight lines. Another concern was navigation within the 3D space. I questioned whether users would find navigating a 3D display as natural as a 2D display. To examine these questions, I implemented a vertical prototype network communication visualization in OpenGL and called it Network Eye GL (Figure 23).

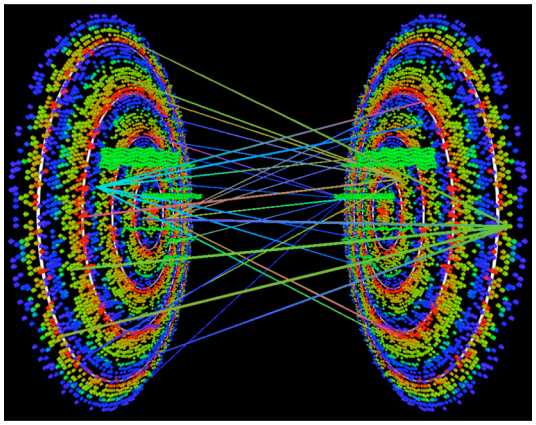


Figure 23: Network Eye GL, an OpenGL vertical prototype of the Network View.

But the network view was not all that the users needed to see. As discussed earlier, the typical intrusion detection scenario was that they would note suspicious communication patterns whether on the network or in log files, and then they would go to the host(s) that were causing the problem and find out what processes were responsible for this activity. Thus, I needed to provide the user community with a view of host activity that could be correlated with the network activity seen in the network communication visualization.

Key Conclusion: Network Eye GL and the Network Communication Diagrams generated more excitement than insight among users. While system administrators liked the look of the tool, using it to locate a particular host was very difficult. People often got lost in the 3D world and sometimes could

not get back to a known position. Users said that while the display could probably handle the amount of network data they typically see, it would be an illegible mass of lines unless 99.9% of it was filtered out.

4.2.4 Host Resource Visualization

The next visualization in the Network Eye framework displayed host processes and their communication resources. It meshed closely with the network communication visualization. The host resource visualization (Figure 24) showed the interaction of applications in the overall network. An application can be seen as a set of processes that use sockets and other resources to operate. The sockets are bound to some port, and these ports may be communicating with remote hosts. The ability to trace a port back to its owning application greatly helps in determining why the communication is happening. Both client-server and peer-to-peer relationships can be discovered in this way, regardless of encryption.

The host resource visualization shown in Figure 24 has three parts or layers. At the top is the set of applications and their associated subprocesses and sockets. Below that is a pixel map of all 65,535 TCP ports on the machine organized numerically from the low-numbered, "well-known ports" up to the high-numbered, ephemeral ports. At the very bottom is a network pixel map showing the hosts that the ports are connected to. This map could be used to connect the host resource visualization to the network communication visualization.

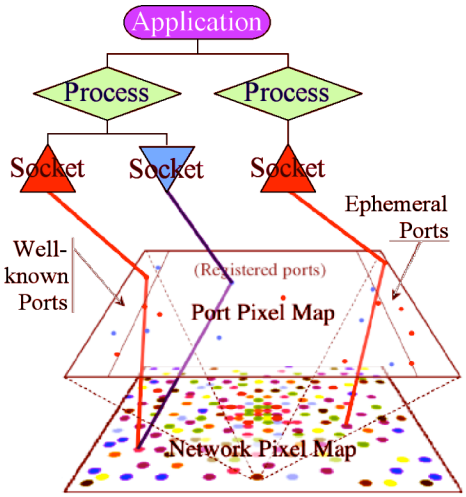


Figure 24: Network Eye's Host Resource Visualization

Obtaining detailed information from a host would require a "tattletale" client to be installed there. This is a reasonable requirement for my users since remote administration tools are common. My users requested that the host-application resource visualization also be able to show files related to each application to make it easier to locate and eradicate unwanted or malicious programs.

4.2.5 End-to-End view

When one or more host resource visualizations are hooked into a network communication visualization, an analyst may use the network communication visualization to spot some communication patterns of interest, select a communication line, and drill down into the resulting host resource visualization to find out what hosts, ports, sockets, processes, applications, and users are responsible. Host drill-down enables quick investigation and reaction to potential intrusions. I call this joined view the end-to-end view of network communications (Figure 25). With the end-to-end view,

users could see from the client processes on one end, across the network to the server processes on the other end. My users were very excited about the end-to-end view but were mostly interested in the ability to see inside monitored hosts from the network and trace communications to the applications responsible for them.

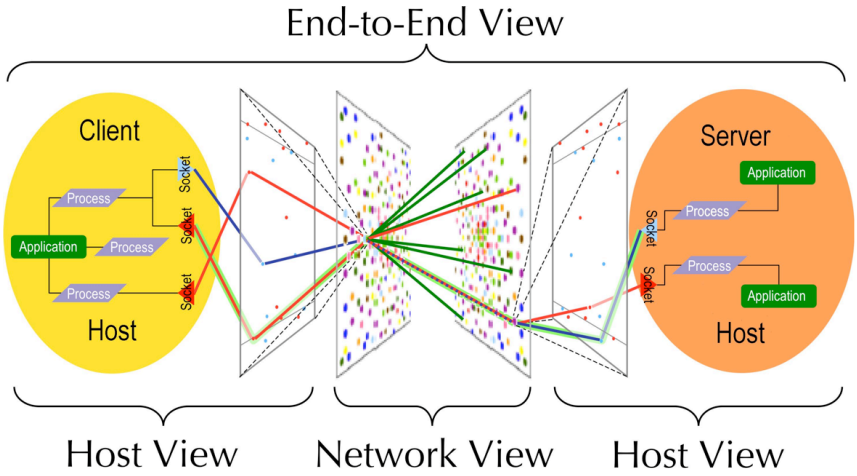


Figure 25: Overview of the Network Eye application.

4.3 High-Fidelity Prototypes

Beyond the original low-fidelity prototypes, I created a number of higher fidelity prototypes that could be used to conduct usability studies on my implementations. The first prototype, Visual Information Security Utility for Administration Live (VISUAL) was a 2D rendition of the network communication visualization. The second, psgraph, was a working prototype of Network Eye's Host Resource View. The third, Network Monitor (NeMo), was a first attempt at looking across the boundary between the network and the host to correlate packets to processes. The fourth prototype, Portall, was an expansion of NeMo that helped finalize many of the concepts I put into the final rendition of my technology.

4.3.1 Visual Information Security Utility for Administration Live (VISUAL)

This section presents VISUAL, my first high-fidelity prototype. I will recapitulate some of the information contained in my paper on the subject[53], but I encourage the reader to study the paper for more information. VISUAL was designed to be a prototype of the network view part of the Network Eye architecture. Robert Ball, Anand Rathi, and Sumit Shahdid the programming for VISUAL as part of their work for a class on Information Visualization. They conducted a usability study of VISUAL with me and published a paper (referenced earlier) at the first annual VizSEC conference. Finally, I brought the prototype to some of the original interviewees (and a few of their associates) to get freeform feedback on the utility of the design.

4.3.1.1 Visualization Design

We used participative design techniques[10] to build VISUAL, tempered with what we could do easily at the time. Given the single semester deadline, we elected not to try a full 3D, interactive network visualization but instead chose to use a 2D display. Figure 26 shows what VISUAL's main window looked like. The main purpose of VISUAL was to show a large number of hosts in a network view at the same time. Many of the administrators I interviewed are responsible for over 100 systems and they must monitor traffic entering or leaving them for potential intrusions. Other visualizations I

found could only handle a single "home" host and about 30 foreign hosts. This was nowhere near enough for many of my users.

Network administrators are interested foremost what is happening on their own network(s). They want to see the impact of the "unsafe'" external Internet on the machines they manage. My premise with VISUAL was that by visualizing communications to reveal the subjects, objects, and duration of conversations, network administrators would be able to identify patterns that may be difficult to detect by textual methods. In fact, we elected to evaluate VISUAL with completely inexperienced subjects to see they could detect important patterns like fan-in and fan-out without any training. Aside from seeing abnormal traffic, VISUAL allowed network administrators to develop an accurate mental model of what was normal on their own network so that they could diagnose problems better.

VISUAL's design followed Shneiderman's visual information-seeking mantra[57], "Overview first, zoom and filter, then details on demand." VISUAL displayed an overview of network traffic for a small to mid-size network. VISUAL also showed a home-centric, internal vs. external perspective of the network displaying each home (internal) host as a small square within a larger grid that stood for the set of home hosts as shown in Figure 26. A user could see which home hosts received connections from a large number of external hosts (fan-out) and which external hosts communicated with a large number of internal hosts (fan-in). Although there are many ways to detect these phenomena, VISUAL showed that a concrete visualization allowed even untrained users to detect these patterns. Furthermore, communications patterns were visible in the context of the total network state, showing relative amounts of activity among external hosts by the size of their markers.

Basic Layout and Features

VISUAL used a predefined list of home hosts stored in a text file. After reading the list of home IP addresses, VISUAL loaded and displayed the network traffic. Among the items that VISUAL displayed were:

- A spatially related grid of home hosts.
- Statically placed markers for the external hosts that communicate with hosts on the home network.
- Amount of traffic exchanged between each external host and the home network.
- Protocols and ports used during the communication.
- Temporal replay of network communication events.

The first display priority was the set of home hosts. The large square grid in the display area represented the home network. Each grid square represented a computer on the home network. The home grid was automatically positioned in an empty area of the display space. The home network could be moved or resized to avoid overlap.

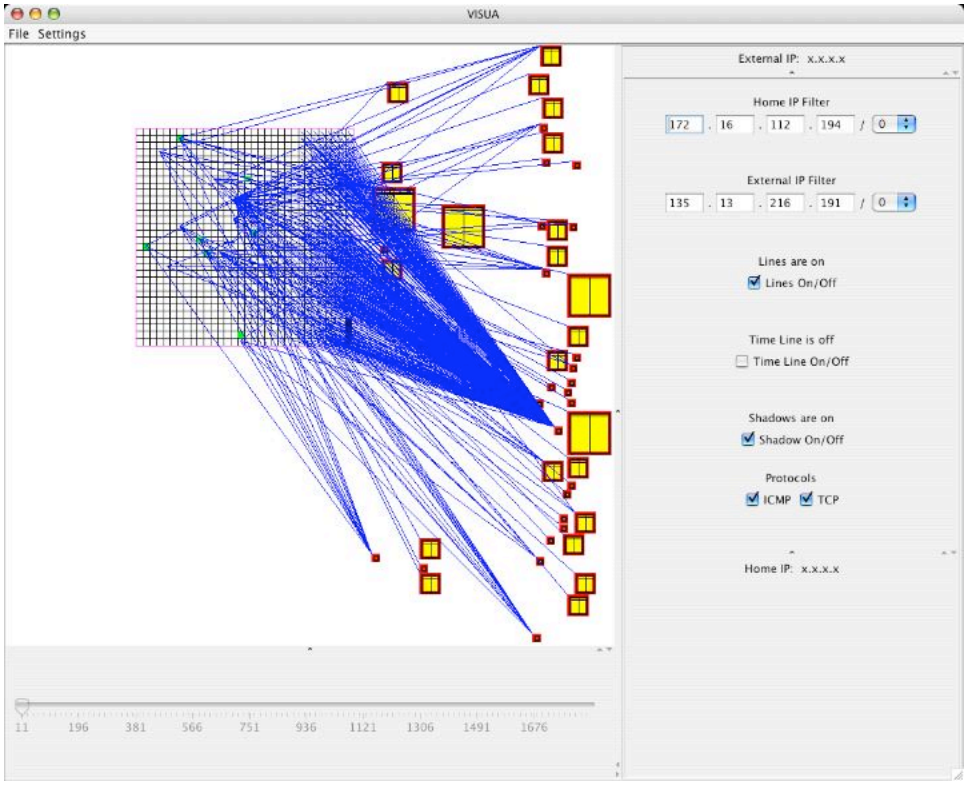


Figure 26: Screenshot of VISUAL displaying 80 hours of network data with a home network of 1,020 hosts, 183 external hosts and 915 communication lines.

The second display priority was to provide markers for external hosts (4 billion possible addresses in IP version 4) that communicated with the home network. I attempted to maintain a constant position for each external host. If a given host appeared in two different datasets, I wished to have its screen position to be approximately the same in both plots. Therefore, the mapping function assigned every external IP address a unique virtual position in the display area. I then used techniques adapted from Keim[58] that helped to display large amounts of spatially referenced data on a limited-size screen display. I used the first two octets (the most significant 16 bits) to determine the *x* screen coordinate and the last two octets (the least significant 16 bits) as the basis for the *y* coordinate.

A drawback of this simple mapping scheme was that IP addresses that were similar (especially those that differ only in the second or last octet) mapped to points very close to each other. If two external IPs mapped to the same space in the display area, I moved the one plotted last to the nearest available position. Depending on the set of IP addresses and the plotting order, some markers could be displayed in different locations at different times. I adopted a heuristic of plotting the initial set of markers in IP address order to somewhat mitigate this problem.

A feature of this mapping was that external IP addresses with the same first two octets appeared clustered in vertical lines. This fact may help network administrators to see subnetwork patterns. Near constant positioning greatly helped users because as the data changed from day to day, the user could find a given external IP in the same relative position. Since computers do not frequently change IP addresses (at least within a tightly constrained range) my simple mapping approach could help administrators detect patterns from session to session. My mapping approach may be able to display up to several thousand external hosts.

The third display priority was to show which home network computers are communicating with which external computers. To facilitate this, I displayed lines from individual internal hosts (grid squares) to the external host(s) they are communicating with. Each line represented a communication of one or more packets but did not indicate how many. The communication bandwidth was encoded as the size of the external host markers.

As mentioned earlier, fan-in and fan-out were evident in VISUAL. Figure 27 (a) shows an external host performing a ping sweep (a method attackers use to discover what computers are on the network). The external host systematically contacted every IP address in a particular subnet in the home network. This visual representation of an attack is easily recognizable by most users, without regard to experience in networking. Figure 27 (b) illustrates fan-out, where several external IPs were contacting a particular computer in the home network. Although this pattern could indicate a denial-of-service attack, it happened to be a public FTP server that external hosts were accessing.

Line color showed the direction of traffic. For example, Figure 27 (a) shows an external host sending packets to many different internal computers without any reply. Unidirectional communication sent from an external host to an internal host was presented in red. Bidirectional communication was shown in blue. Communication that left the home network without receiving a reply appeared in green. In VISUAL's parlance, established TCP traffic was always considered bidirectional. UDP, ICMP, and most other protocols may be unidirectional unless the recipient of the traffic also responds. In Figure 27 (a) the red lines indicated the target hosts did not reply. However, there were a few blue lines showing some hosts did reply. Assuming the policy is not to reply to ICMP from external hosts, these few may have replied because they had been compromised.

VISUAL emphasized internal to external and vice-versa traffic because administrators I interviewed indicated that they generally monitor the internal-external traffic until they notice a potential security hazard. Then they turn their attention inward to locate and eradicate problems within their own bailiwick. To view internal traffic, VISUAL provided an internal-internal mode where all internal hosts were plotted twice: once within the home grid, and a second time outside it as if they were external hosts. In this view, external hosts are not plotted. In this way, users could quickly identify which internal hosts were responsible for which traffic.

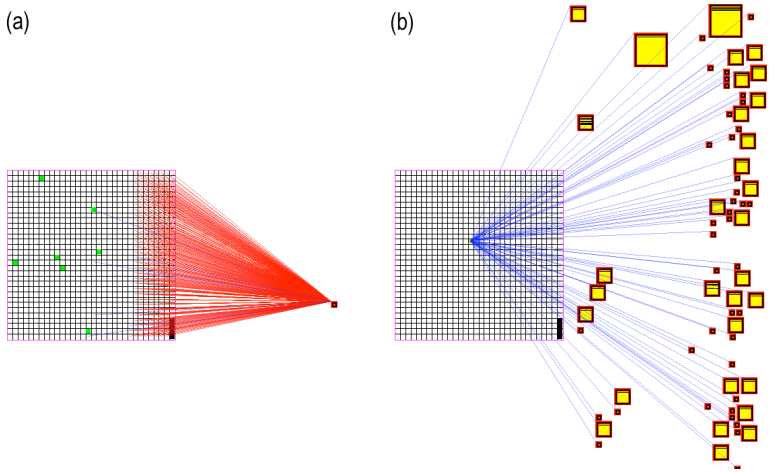


Figure 27: VISUAL examples of (a) Fan-in: an external computer performing a ping sweep on a subnet in the home network, (b) Fan-out: many different external computers are downloading information from a server on the home network.

The fourth display priority was to show the rough amount of traffic each external host was responsible for during the observation time period. I sized the marker of every external host in proportion to how many packets it received/sent during the time frame compared to the other computers that communicated with the home network. I elected to use a discrete scale of only three marker sizes to make differences more noticeable. I grouped observed traffic levels from each host into three clusters via the K-Means clustering algorithm. The largest markers represented hosts in the cluster that contributed the largest amount of traffic, while the smallest markers represented hosts in the cluster that contributed the least traffic.

The fifth display priority was to visualize what ports and protocols each external host uses to communicate. To show each port number on the screen space as an icon or as text for each square would clutter the screen space. Instead I displayed the destination and source ports as horizontal lines within the external host's marker (see Figure 28). There are 65,535 possible TCP ports that a computer can use, so I scaled this range to fit within the marker to show at a glance approximately how many ports an external host was using. VISUAL did not differentiate between TCP and UDP ports. The home network computers did not display the port visualization except in internal-internal display mode.

After seeing the overview, the user might focus on a few internal computers of interest. A user may select a one or more internal host grid squares (Figure 29), to see only the traffic that occurred with those particular computers, filtering out all other data. External host markers may also be selected to filter out communication lines by any other external host. A user may select host markers by clicking on them or by dragging a selection rectangle that touches all of them. With shift- and control-clicking, the user may select complex combinations of host markers, or she may filter a particular set of addresses by using a Classless Internet Domain Routing (CIDR) bit mask filter.

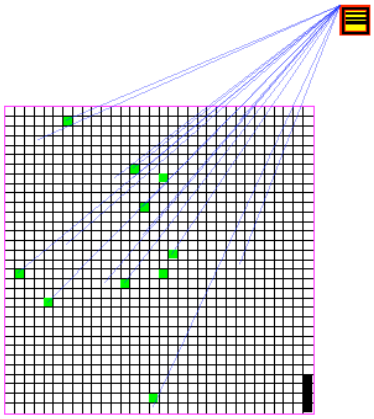


Figure 28: A single external host using multiple ports to contact hosts on the internal network.

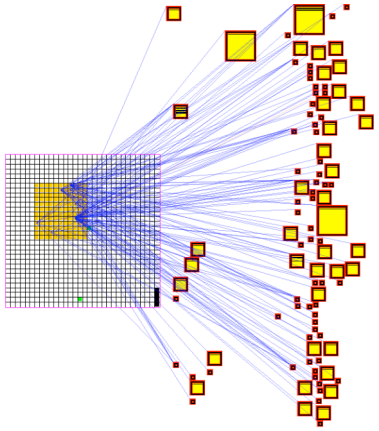


Figure 29: Multiple selection. Only traffic for the selected (orange) home computers was displayed.

Extended Features and Abilities

By default, all communications for the analysis period were shown in the overview at once. To see the temporal relationships of the communications, the user had to activate the time line (Figure 30) to limit the display to show only the network activity that occurs within a sliding, one-second window.

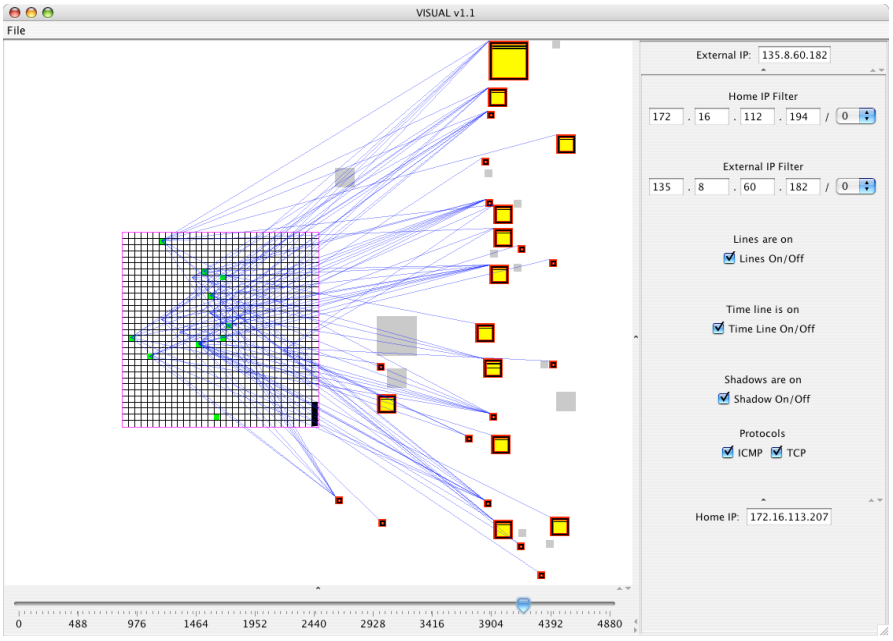


Figure 30: Time line example using shadows. The yellow squares represent active external hosts during the selected second. The light-gray squares represent inactive external hosts that have been active in the last 200 seconds.

The timeline helped users quickly track the chronological flow of events. I used *shadow markers* showing what external host markers were recently active to mitigate "change blindness"[59]. As [60] posits, I expected it would be difficult for the human mind to keep track of which hosts were recently active without these shadow markers to provide temporal context.

Filtering controls were located in the control panel on the right side of the application. There were check boxes for toggling all communication lines or just those for a particular protocol (e.g., TCP, UDP, ICMP, etc.), the time line, and the shadow markers.

The user could control the physical size of the home network grid and the sizes of the external host markers reducing them to a single pixel or allowing them to overlap one another. By minimizing the size of the markers, the display could handle nearly 10,000 markers at a time, but if enough traffic were shown, communication lines would likely occlude too much of the diagram to make it useful.

The user could get the following details about the computer represented by a marker on demand:

- The host's IP address
- The IP addresses of all the computers that the selected computer communicates with
- The TCP/UDP ports (both source and destination), the host uses to communicate
- The percentage of the overall traffic this particular computer contributes to the overall dataset within the analysis time period

4.3.1.2 Usability Study

The objective of the usability study was to determine whether inexperienced users could identify important communications patterns without training. For my usability study, I asked users to compare two or three datasets and identify behavior that was markedly different between the datasets.

I conducted my usability study with eight graduate and undergraduate university students. During each session I had each user answer some biographical questions to determine their experience with computers and network security. The users described themselves on a range from "power users" to

"occasional users" of computers. I purposely excluded network or system administrators from my study because I wanted to test untrained users. When asked about the interface, all the users characterized VISUAL as easy to use.

I explained to each user how VISUAL works then allowed them to become comfortable with VISUAL for about five minutes with a training dataset that did not have any abnormalities in it. Once they were familiar with VISUAL, I presented them two testing datasets and asked them to:

1. Describe anything striking about this dataset.
2. List the IP addresses of four external hosts that appear to be involved in normal (repeated, frequent in time communication with the home network.
3. List the IP addresses of four external hosts that only communicate from time to time with the home network.
4. List the IP addresses of four home network hosts that make the largest number of connections to external hosts.
5. List the IP addresses of four external hosts that connect to the largest number of different home network hosts.

All datasets came from the same network[61]. The first dataset was a training set that contained only normal activity. The second dataset that the users were shown contained large amounts of data compared to the first dataset, but was still normal for the network. The third dataset had normal network traffic except it contained a ping sweep (see Figure 27 (a)) and was slightly smaller than the second dataset.

Every user was able to quickly find the same set of abnormalities in the data. They all used different words, but their answers agreed. Every user was able to quickly focus on the ping sweep as abnormal. Not all of the users knew what a ping sweep was, but they were still able to see that it was a different traffic pattern than the rest of the data presented.

Users were also able to quickly identify the following patterns:

- The external hosts that communicated the largest volume of traffic
- The external hosts that communicated most frequently with the home network
- The internal computers that communicated the most with one another
- The external hosts that communicated to the most distinct internal hosts (based on number of connections)

Users had difficulty using the timeline to identify external computers that I described as "only communicating from time to time" with the home network partly because I did not define "from time to time" quantitatively. Users had difficulty characterizing intermittent communication and identifying hosts involved in it. Another problem users experienced was that communication lines occluded some host markers in the overview window. Making the lines translucent so that the host markers showed through solved this problem.

One user said VISUAL made it "easy to make sense of data and see general trends." This user said VISUAL would be less usable for "fine-grained" view of network traffic data. This observation fit my intentions well for a computer security visualization and underscored the importance of providing drill down to the packet level for analysis purposes. The usability test successfully demonstrated that a traffic pattern visualizer such as VISUAL can provide insight into network traffic data without requiring any training.

4.3.1.3 Shortcomings of VISUAL

VISUAL was a proof of concept visualization of the network view part of Network Eye. However, when it was presented to users, I learned that they would use it to identify suspicious communication patterns and find the hosts that were causing them. Then they would go to those hosts and search for processes that were causing the problems. I realized that network visualization was not enough. Administrators needed to know what processes were responsible for the communications streams seen on the network to diagnose suspicious events.

When I presented our VISUAL paper, a new community was forming for computer security visualization. At the workshop, I saw many other network views, some of which would suffice for my purposes. I then realized that I needed to shift focus away from network visualization because many people were already producing fine high-density visualizations of network traffic. My new focus would be on the critical piece that made the end-to-end view possible, bridging the host/network divide. To do this, I had to work on host resource visualization and on an infrastructure for correlating traffic on the network with processes on the monitored hosts.

4.3.1.4 Key Conclusions

VISUAL showed me that the network view, while important, was not the only thing people needed to see. Many others were working on network visualization, but no one seemed to be visualizing what was happening inside the host or how that related to activities seen on the network. Another problem with VISUAL was that I was trying to lay out a 3D graph in 2D space. This is because I was plotting connections between a 2D space for the home hosts and another 2D space for the foreign hosts. My experiences with Network Eye GL (section 0) showed me that users found 3D visualizations interesting but difficult to navigate. To create a usable security monitoring product, I needed to move back into 2D space. This implied showing the home hosts and the foreign hosts along separate 1D continuums and connecting them in 2D space. Another advantage of the 2D approach was that it tied into the 2D host resource visualization naturally.

4.3.2 Host Resource Visualization: psgraph

Psgraph was a prototype host resource visualization that gathered its data from Unix utility programs and presented a graph showing a tree of all processes that had open sockets. Figure 31 shows a screenshot of psgraph where the blue parallelograms are processes and the green, yellow, and red ovals are open sockets. Psgraph did not attempt to correlate network flows to the open sockets; it only reported programs where a socket is open and whether that socket was connected to an external machine or not.

Psgraph worked by correlating the output of the *ps* (process listing) and *lsof* (list open files)[24] Unix utilities. First, psgraph spawned subprocesses to get information simultaneously from both these programs. It used the *ps* output to construct a process tree of all processes running on the system. Psgraph used *lsof* to find all the TCP/IP sockets that are open, what process they belonged to and what remote host (if any) they were connected to. Then psgraph pruned the process tree of all processes that had no descendant that owned a TCP/IP socket. It used the tree structure to build a graph file as input to the GraphViz[62] processor. GraphViz produced a postscript output file similar to Figure 31 that psgraph displayed on the console using ghostscript[63].

Psgraph displayed only processes that have open TCP/IP sockets and all their ancestor processes. In Figure 31, we see that the *init* process had run the *Window Server*, and it in turn had started four processes: two *Microsoft* processes, an instance of *X windows*, and *Firefox*. The *Microsoft* processes each had two open sockets connected to the local machine only. I colored local-only connections green because they are relatively safe.

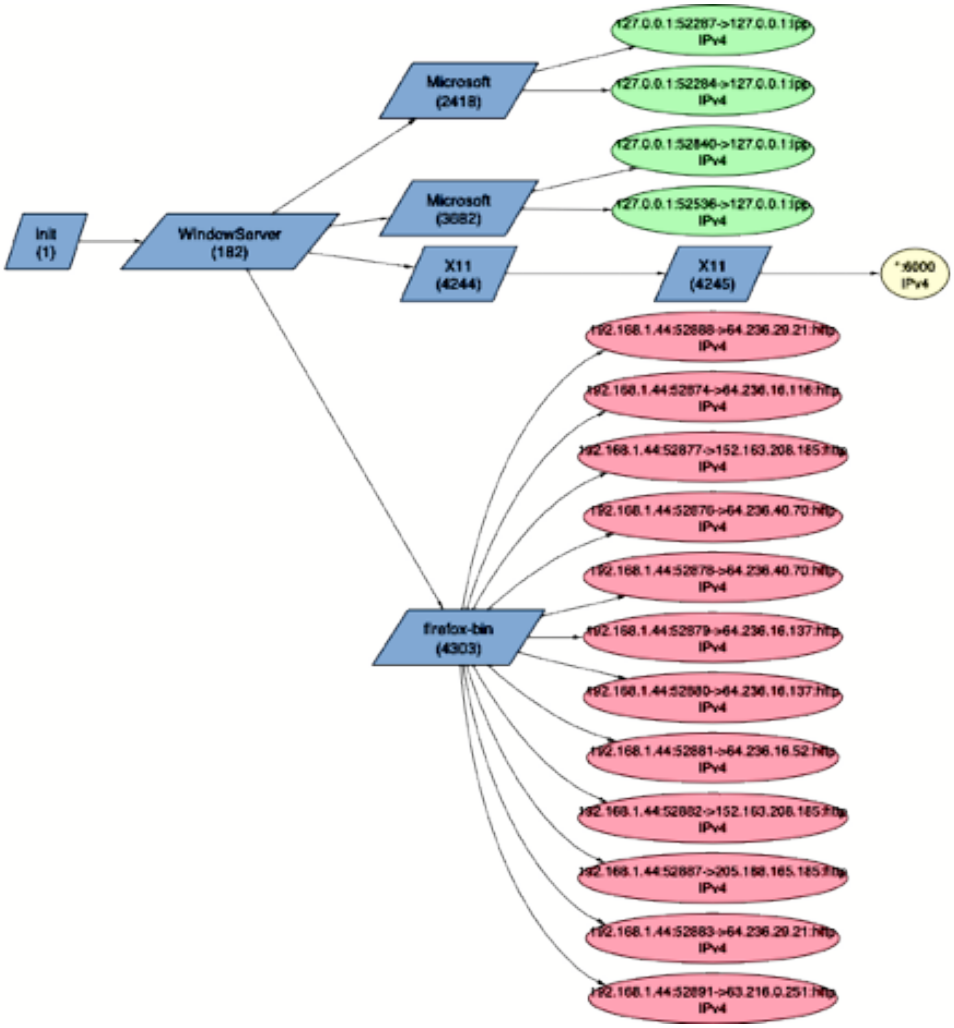


Figure 31: Screenshot of psgraph output showing that application Firefox has several open connections.

The *X Windows* server had spawned another process that was listening on TCP port 6000. Because this socket was listening, it could, as far as psgraph could tell, have been exposing a service to the network. Because of the potential danger that would poses, I colored listening sockets yellow (even though a firewall may prevent these connections from taking place).

The *Firefox* web browser, on the other hand, has 12 sockets open to remote sites. I colored actual connections to foreign machines red because of the potential danger they represented. Red did not indicate anything about the actual machines they were connected to other than the fact that they were remote. In this case, *Firefox* is browsing www.cnn.com.

An associate of mine and I had an interesting experience with psgraph happened when we installed it on his machine to try it out. When he ran the program it showed a remote connection to a Samba file server on his machine. He did have the Samba service running, but he didn't think anyone was using it because he hadn't publicized the service and was just using it to share files with friends. Besides, he had only been plugged into the network for about 20 minutes anyway. Figure 32 shows what xtraceroute revealed to us about the origin of his uninvited file sharer.

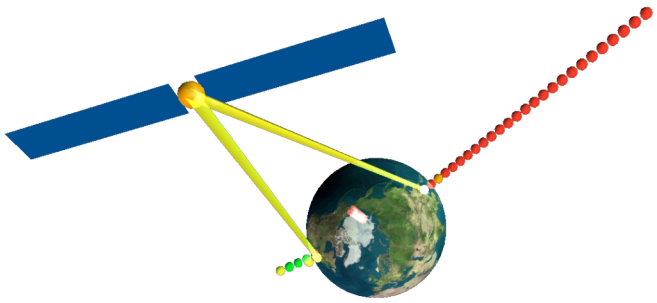


Figure 32: The route xtraceroute displayed when we traced my associate's Samba invader. Unknown to him, a machine in China was accessing files on his laptop in Virginia.

Key Conclusion: Psgraph showed that even a limited view of the host activities could give useful information to security professionals. Text-based tools such as netstat made it hard to pick out unusual activity, but portraying this information as a picture made spotting anomalies easier.

4.3.3 The Network Monitor: A First Attempt at Packet-Process Correlation

As mentioned earlier, when I looked into how system administrators detect and investigate potential intrusions and other problems, I found a common process. First, the administrator becomes aware of an anomalous communication pattern. She may find out about the pattern by looking through log files, via an alert from an Intrusion Detection System (IDS), or (quite commonly) from a colleague whose machines are being pummeled by one of hers. Once she traces the communications to an end-point host, the administrator will look at the processes running on it determine whether one or more of them are malicious. The fundamental problem with this approach is that it requires a human to mentally correlate two sets of data: the communication patterns that first indicated the problem, and the set of processes that may be responsible for generating the patterns.

The Network Monitor (NeMo) was my first attempt to automatically correlate host and network data. NeMo was coded in C# using the Microsoft .Net framework[64] by Nipun Jawalkar, an undergraduate research assistant. NeMo used the WinPcap[65] packet capture library and .NET's IPHLPAPI[66] to capture host process and socket information. A screenshot of NeMo's main window is shown in Figure 33.

NeMo was able to correlate network activity with host processes on a gross level using two Windows system calls that were undocumented at the time NeMo was written: AllocateAndGetTcpExTableFromStack[67] and AllocateAndGetUdpExTableFromStack[68]. These two calls return the PID and any associated communication ports from a given process. NeMo used this information to create two four-level tree data structures containing a root, the set of hosts, the processes running on each host, and the active connections for each process. One tree was for clients (who initiate connections) and the other was for servers (who respond to connections). The trees were connected at the leaf nodes to model all communications between client and server processes seen by the monitored machine.

Using only this information, NeMo is no more than a glorified netstat. But I incorporated information from tcpdump to fill in the gaps where processes might have been missed between polling cycles. Using the WinPcap library, NeMo traced every packet it captured to a connection in the two-tree connection model. If no current connection could be found, NeMo added a new entry to the connection model without specifying processes or sockets. In this way, NeMo showed all the

connections it saw, but it did not always know the processes and sockets for all of them. NeMo also decoded and stored certain fields from the packet header for offline viewing and connection summary information.

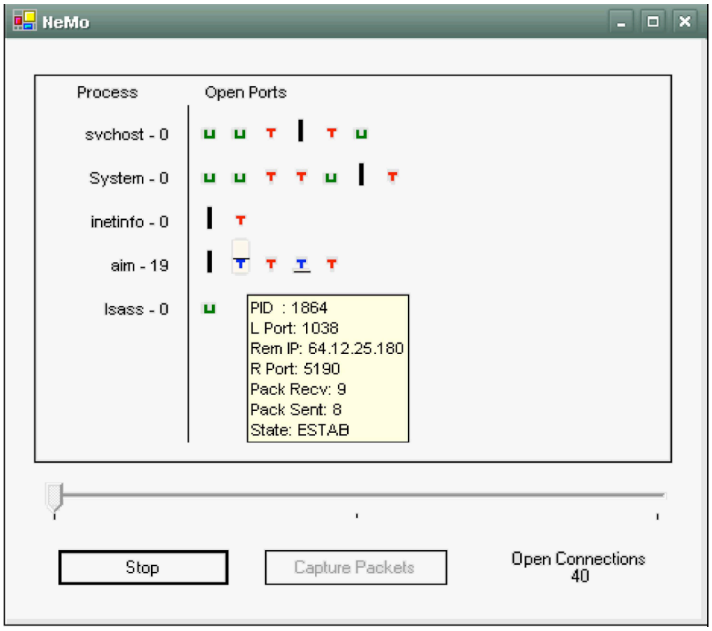


Figure 33: Screenshot of NeMo's main window

4.3.3.1 NeMo's Shortcomings

One problem with NeMo's approach was that the packet-process correlation was very loose. NeMo could only poll the kernel once a second for connections, so rapid connections could easily be missed. The other side of this issue was that the packet capture engine interpreted every packet as a connection, even if it could not find a matching process. This caused stray packets on the network to look like actual connections when the monitored machine may not have even accepted the packet. NeMo was not really a visualization at all. Rather, it was a tabular layout of active processes and ports on the local machine and could at most be considered a dashboard. Finally, NeMo's layout did not lend itself to displaying connections from more than one monitored machine at a time, so it would not have been capable of displaying a true end-to-end view.

4.3.3.2 Key Conclusions

NeMo showed that crude packet-process correlation was possible, but the layout was all wrong. What was needed was a more visual presentation that took advantage of information visualization techniques.

4.3.4 Portall: Visualizing Packet-Process Correlation

I designed my next visualization prototype, Portall, to improve on NeMo's correlation by *visually* correlating host process data with network communications patterns for investigative analysis. I conducted a usability evaluation of Portall to determine which parts of the problem a visualization could address by itself, and which parts required additional infrastructural support.

Portall used the same basic data-collection infrastructure with some incremental improvements. The main difference is that Portall was designed to visualize connection data from some number of remote hosts rather than simply to monitor a local host. Portall got its data from a database that could

be located anywhere and accessed via secure SSH or SSL connections. Portall could visualize current events without logging to a database, but without the database, it could not review historical events.

Portall was designed primarily for investigative analysis (when there is a suspected intrusion) and secondarily for informative monitoring (when no intrusion is suspected, but the user is preventively looking for signs of one). Although it could be used for forensics, I did not design Portall as a forensic tool. Paul Muessig, a senior undergraduate research assistant, did all the programming for Portall according to my design and under my direction.

4.3.4.1 Portall Implementation

Portall's purpose was to enable its user to discover the meaning behind the data rather than creating yet another automated intrusion detection system. I intended Portall to reveal the overall communication situation in a single screen, leaving the intelligence and analysis to the human analyst. Artificial intelligence in any form was intentionally left out of the design. There are many intelligent tools available to my users, and I did not intend to try to persuade them to abandon their favorites. By presenting information visually, Portall relied on recognition rather than the slower and more cognitively challenging task of recall. The need for real-time analysis led me to design displays with preattentive features to speed pattern recognition.

A typical usage of Portall was investigative analysis after an IDS has raised an alert about a potential security incident. A user could conduct initial investigation with Portall to get a rapid and accurate mental model of what was happening on the affected machines. Then he may use more specialized tools to investigate in greater detail. Portall could be used for informative monitoring by running it a few times a day to check the security state of critical machines. Running Portall's data collection processes continuously could reduce production system performance; thus, I envisioned users running it only periodically.

Portall's Displays

Figure 34 shows what Portall displayed on a malware-infected system, Dudette. Close inspection revealed some suspicious client processes (salm, wmiprv, msmon, Points Manager) making external connections. Also, an external client machine (128.91.76.246) was making a connection to a service on Dudette (although Dudette was supposedly not providing any services).

The main window shows a client-server layout of communications from the monitored machines' perspective. On the left are the client machines, processes and ports. On the right are the server machines, processes, and ports. Client hosts are those with processes that initiate communication with other processes (via a TCP SYN packet). Server machines are hosts whose processes accept communications from clients (via a TCP SYN-ACK response). If the same machine has both client and server processes, icons for the host will appear on both sides, with its client processes on the left and its server processes on the right. User interviews led me to believe that this direct modeling of TCP's function would be clear, although my usability study revealed an unexpected reaction.

Paul and I selected neutral, complementary colors and subtle shading gradients for the host and process markers to de-emphasize them. We selected contrasting, primary colors for the port boxes and connection lines to draw the user's attention to the communication activity.

In Figure 34, item (1) points to icons for the local host. The upper icon represents the monitored machine, and its label is the host's DNS name. The lower icon is a termination point for traffic that seems to come from the local host but cannot be traced back to any visible process on the machine. Its label is the IP address "0.0.0.0." I consider any connections from this icon highly suspicious. Item (2) points to a remote machine that is a client of a server process on the monitored machine. This may be suspicious if no public services are running.

Item (3) points to a column of client process icons. These icons are connected to the machine where the processes are running by a black Bezier curve. Item (4) points to communication lines that link clients to servers and symbolize a TCP connection. Portall makes communication lines for TCP traffic only. The line and port box color indicates the TCP connection state. Item (5) shows the list of server processes. On unmonitored hosts, I can give no information about such processes, so I label them "?". I included these icons because they prevent confusion caused by lines that otherwise could cross under the icons of known processes.

Item (6) shows the icons for the monitored host on the server side. Again, one marker has the DNS name of the monitored host for a label, while the other is labeled "0.0.0.0." As on the client side, the latter icon is for connections that are apparently to service ports on the monitored machine but that cannot be correlated to any visible process. Connections to this marker may indicate attack traffic, misdirected traffic, or possibly hidden services running on the monitored machine. Item (7) points to a column of icons for remote machines seen on the network as traffic destinations.

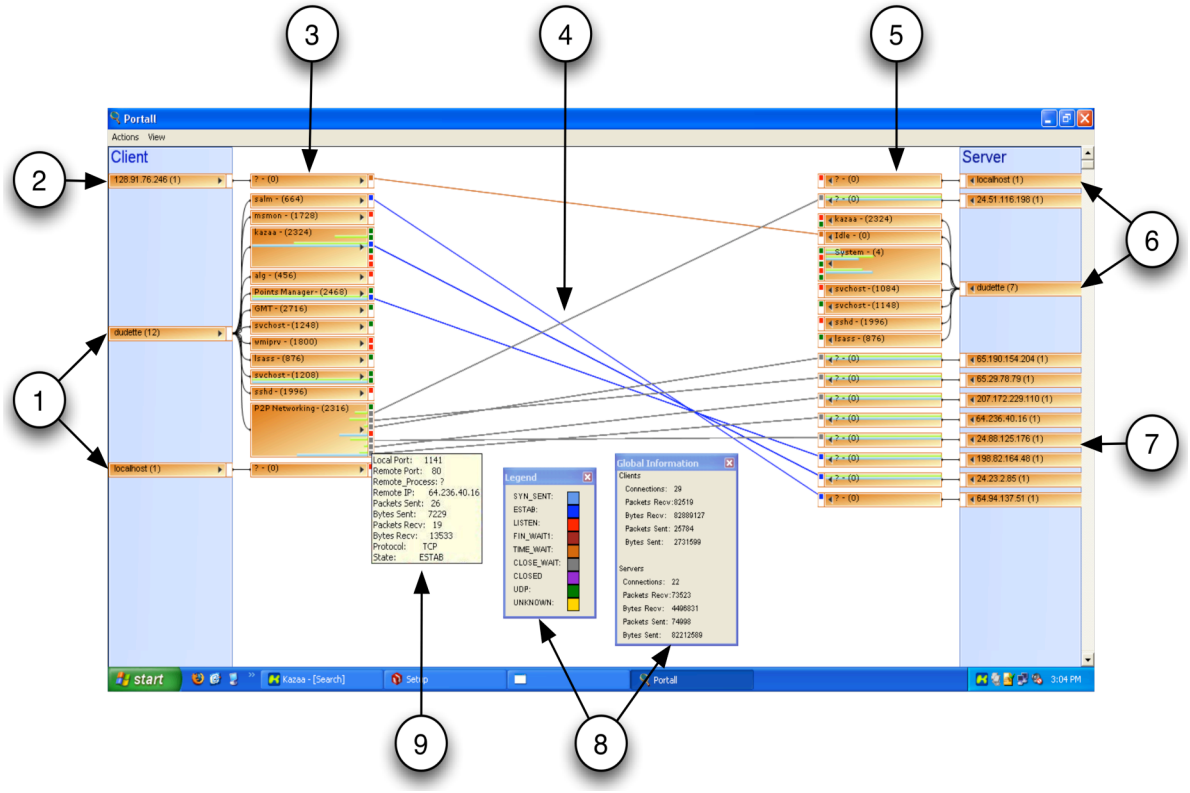


Figure 34: Portall screenshot: (1) The monitored host on the client side, (2) an external client host, (3) client processes, (4) client-to-server connection lines, (5) server processes, (6) monitored host on the server side, (7) external server hosts, (8) information windows, and (9) pop-up connection details.

Item (8) shows the color legend and global information floating windows. The legend shows the colors used to indicate the connection's state. Seven colors are used to indicate the TCP state, one color indicates a UDP connection, and a final color indicates other communications. The global information box shows the number of client and server connections, and the packet and byte counts of all the connections monitored in the current session.

Item (9) is an information box that pops up when the mouse hovers over a port. The box tells the connection's local and remote port numbers and IP addresses, the packet and byte counts in both directions (since the communication was established or monitoring began), the protocol, and the connection state.

Not shown above is the packet-dump window that contains a table with packet-header fields from a connection. Right clicking on a port box shows all of the intercepted packets in the packet dump window for low-level analysis. I have also omitted the timeline window that allows users to visualize traffic and processes logged to the database at some point in the past via a horizontal slider bar.

Users can visually highlight communications lines of interest to bring them forward when there are many occluding lines, by simply clicking the communication lines, or their ports, or their process icons.

Figure 35 shows the detailed features of the process icons. Process icons contain a lot of information: Item (1) is the process executable name and the process identifier (PID) number, item (2) shows bars indicating the relative activity of each port (green for incoming, blue for outgoing, and grey for overall), item (3) shows the connected communication lines, item (4) shows a port box (this one is a UDP port), and finally, item (5) shows the control that the user can click to show or hide the port list. Hiding the ports vertically shrinks the process icon and is useful if a process has many open ports that are not of concern. The icon's shading indicates whether the port list is shown or hidden.

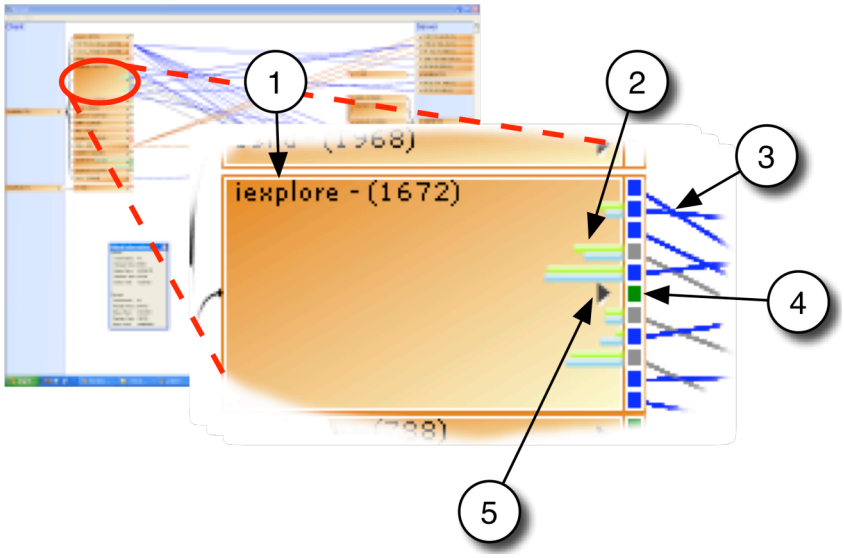


Figure 35: Detail of a process icon: (1) process name and process identifier (PID), (2) port activity bars, (3) communication lines, (4) port box, and (5) expand/reduce control.

To help highlight changes to the display, new screen elements (connections, hosts, and processes) were colored a highly contrasting green for their first display cycle (a second or less). Recently active ports appeared briefly as outlines, and newly opened ports appear double-sized for the first cycle.

Architecture

Portall was designed to be a distributed application to visualize data collected from remote sources. Figure 36 shows the distribution of these components I used on my three-machine network for the usability evaluation.

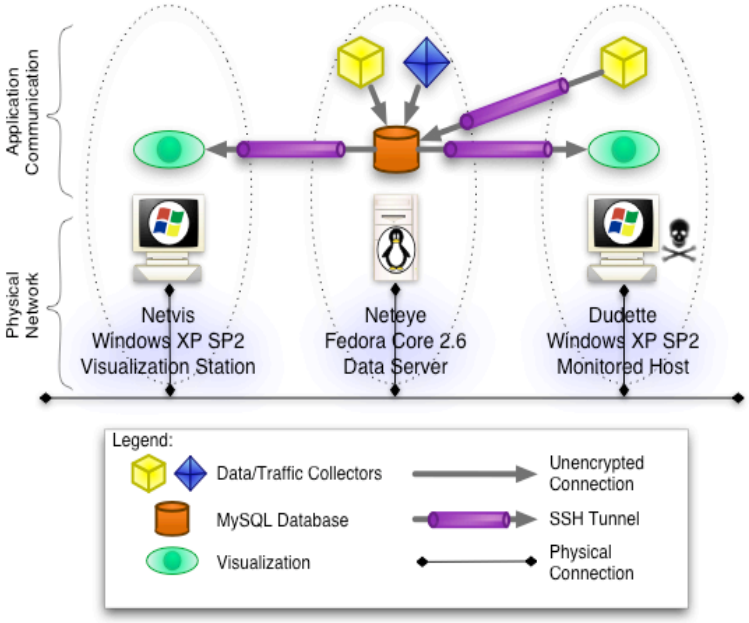


Figure 36: Physical and application layout for the Portall demonstration system.

The three machines on my demonstration network each had a primary purpose:

- Netvis: This Windows XP machine was my primary visualization platform for the usability evaluation, and I verified that it was free from malware. I ran Portall on this machine, but it could be used to visualize activities of any of the three machines.
- Neteye: This machine was a Red Hat Linux (Fedora Core 3) machine running a MySQL server, a traffic collector, and a process data collector. Neteye's primary job was to provide data for the visualization platforms.
- Dudette: This host was running Windows XP and a large number of other malicious and questionable processes. Dudette's primary purpose was to serve as a monitored host although technically, it could visualize data from any of the machines.

I scanned Dudette with a variety of antivirus and spyware/ad-ware detectors and found that it had at least the following known malicious processes running on it: W32.Spybot.Worm, wpma36c.exe (unknown malware), W32/Rbot-WM worm, a variant of the GEMA.D trojan, and several executables classified by Norton Antivirus as a miscellaneous Backdoor.Trojan. NoAdware v3.0 found 180 files contributing to the following ad-ware installations: 180Search Assistant, AutoUpdater/Envolo, ISTbar/Powerscan, PeopleOnPage, AvenueMedia (DyFuCA), CashBack, NCase, Adware.SyncroAd.

Dudette was obviously a very distressed machine. In fact, the CPU was typically at 100% utilization, and frequently one could not make any legitimate outgoing connections from it because it was so "busy." Unfortunately, problems with the tremendous amount of malware running on Dudette prevented me from reliably connecting to the database on Neteye. Thus, in the usability study I did not attempt to show remote visualization.

Scalability

One of Portall's key design goals was to present a single-screen overview of the communications and processing activities of one or more networked hosts. Thus I had to consider how well the visualization approach and architecture would scale as numbers of machines, connections, and

processes increased. Portall did no aggregation, so there was a marker for each machine and process and a line for each connection. Given a typical 1280x1024 pixel monitor, there was vertical space for just 39 processes and hosts on each side of the display. A single process with multiple open connections (such as the Kazaa example shown in Figure 37) easily used up this space.

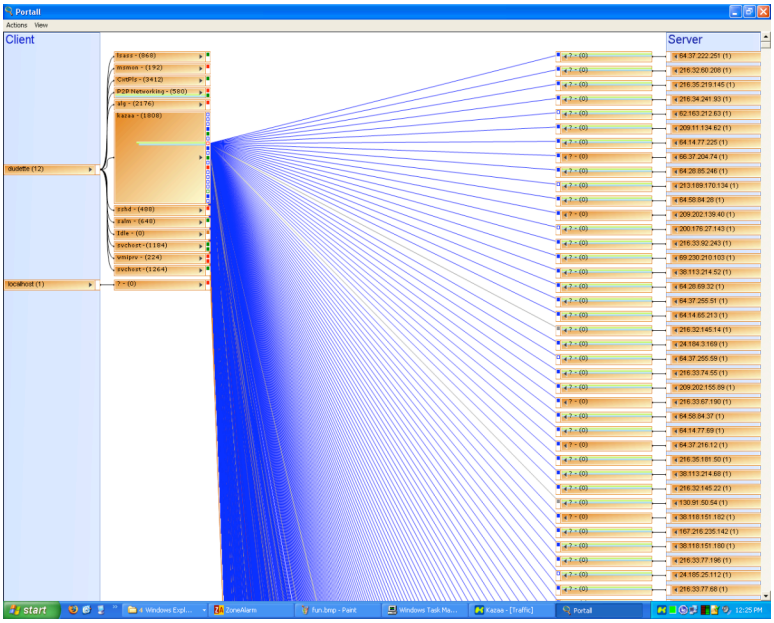


Figure 37: Remote connections from a single Kazaa process can exhaust Portall's display area. This shows the need for aggregation strategies.

Portall could visualize data from a single host and network connection, but the goal was to monitor multiple machines and potentially multiple networks from a single display (see Figure 38). My decentralized deployment design called for an overlay network of SSH connections linking monitored hosts (that serve internal host data) and network monitors (taps, etc. that serve connection data) to an aggregation engine on the local network. These aggregation engines in turn served data to local and remote visualization stations for monitoring. SSH handled the authentication, confidentiality, and data integrity issues for the system.

4.3.4.2 Portall Formative Usability Study

I performed my Portall usability study with a small group of experienced users. My main research questions were:

1. Does correlating network traffic with host process data provide useful insight to system administrators for investigating anomalies?
2. Is my visualization capable of presenting this correlation data in an understandable and effective way?
3. What kinds of filtering do system administrators need to investigate potential intrusions?

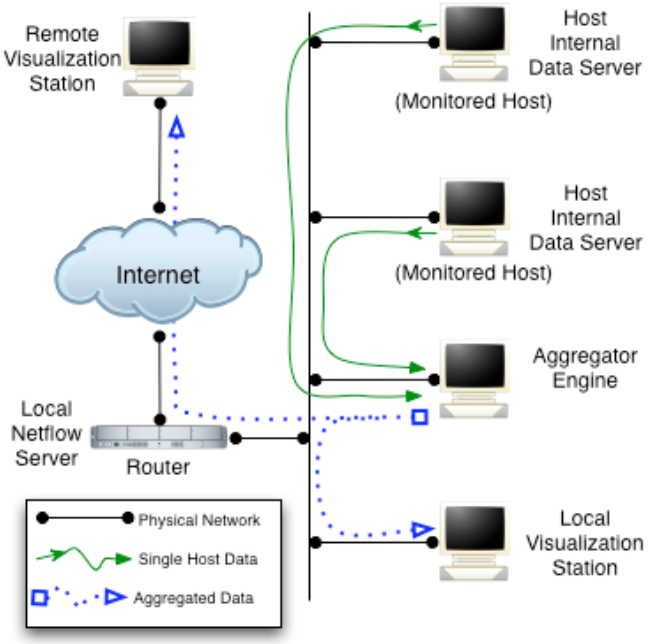


Figure 38: Portall, as part of the Network Eye framework, was designed to eventually display the activities of multiple hosts and network connections remotely via a hierarchy of data servers, aggregation engines, and visualization stations.

To answer these questions, I conducted 45-minute usability evaluations of Portall with five subjects visualizing live data under realistic circumstances. I consider this a formative evaluation because I wanted detailed feedback to improve my design, and I did not yet have sufficient information on the kinds of filtering tools needed for an actual investigation.

Study Subjects and Methods

I chose five system administrators known to be experienced in computer security to test my software and provide recommendations. I started with a venire of 17 administrators whom I selected because of experience I knew they had or because their posts to a campus-wide system administrator e-mail list indicated such experience. Five subjects volunteered from this group. Three of my subjects were from the group of system administrators who expressed tool needs in my initial requirements-analysis study; two were new to my research. The study subjects were offered no compensation. All the subjects had worked as professional administrators for between 8 and 32 years (mean 18.2, stdev 9.8). I scored the subjects' expertise using a weighted average of five metrics (each on a five-point scale). I asked subjects to rate themselves on their expertise at Windows, netstat, and tcpdump use, and on their ability to detect intrusions. I asked how often they used security awareness tools to investigate an anomaly as an indicator of how often they might need to use Portall. The lowest frequency was 1-2 times per month, and the highest was 4-5 times per day. From the composite expertise scores, I considered three subjects to be experts and two to have moderate expertise.

My usability study consisted of 28 questions, starting with a structured biographical information section, and concluding with a semi-structured evaluation. For my study, I prepared two equivalent workstations with Microsoft Windows XP service pack 2. Neither machine was running any public services. The first machine, Netvis, was loaded just for the study and was clean of any malware to the

best of my knowledge. I obtained the second machine, Dudette, from a careless user and found it to be rife with malicious software of all sorts. After allowing the users to become familiar with the visualization on Neteye, I showed them the same visualization on Dudette. The interviewees were not told upfront that Dudette was hacked. I wanted to see how much they would notice by themselves first. If they gave up after a reasonable amount of time, I revealed the evidences of problems that Portall was showing about the machine (see Figure 39) and discussed them with the subjects. Thus, I planned that even users who were unfamiliar with Windows would be able to have a baseline of normality to compare with.

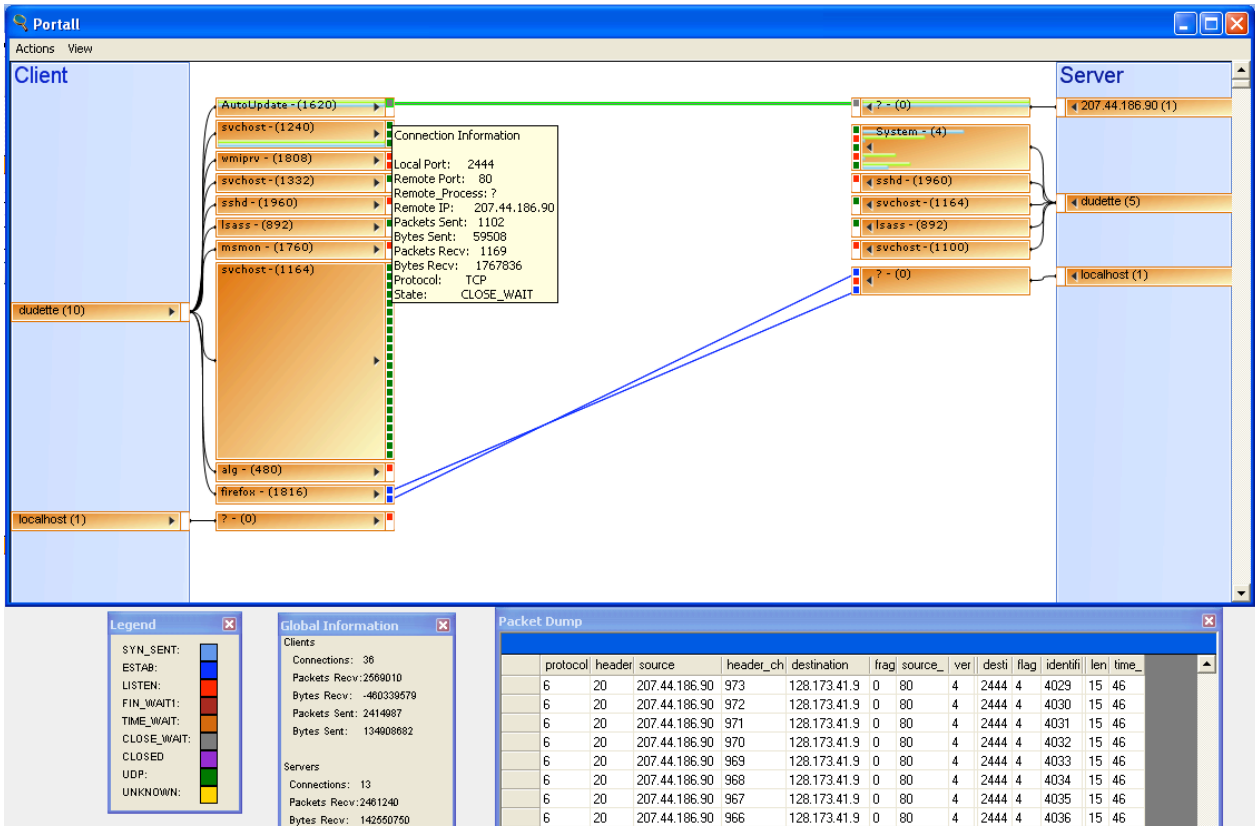


Figure 39: The uppermost connection shown in this screen shot (a grey line highlighted in green) shows a half-open connection. Data was passing over the connection as shown in the Packet Dump window (bottom right) although it appeared to be closed from the client's side. I believe this was a covert channel (innocuously named "AutoUpdate"). This was somewhat subtle, and users did not notice it independently.

After showing the subjects the visualization on both machines, I asked them questions in four areas: (1) initial observations and impressions of the visualization, (2) preferences for how this visualization should operate, (3) reflection on how the visualization might be improved, and (4) interest in the visualization (how they might use it in their work, and whether they would be willing to evaluate follow-on prototypes).

Discussion and Analysis

From the evaluation, I gathered 69 issues: 19 positive comments, 22 negative comments, and 28 enhancement requests. I have summarized the major issues in Table 6 (positive), Table 7 (negative), and Table 8 (enhancements). I classified each issue as one of the following:

- *Fundamental* (F): An essential issue with the visualization design (14 positive, 7 negative, 16 enhancements).
- *Incidental* (I): A prototype implementation issue not fundamental to the visualization design (1 positive, 11 negative, 8 enhancements).
- *Experiential* (X): An issue that was dependent on the user's experience or knowledge (e.g., familiarity with Windows, etc.) (4 positive, 2 negative, 0 enhancements).
- *Aesthetic* (A): Issues of personal taste (0 positive, 2 negative, 4 enhancements).

Table 6: Major positive issues.

Type, # of users	Issue Reported or Comment Made
F, 5 of 5	Users said that Portall's correlation of traffic and process data provides useful insight.
F, 5 of 5	To get similar insight users said would require mental integration of information from several other tools.
X, 5 of 5	Users would use Portall for ambient monitoring of machines or networks.
A, 4 of 5	Made an unsolicited comment that the visualization was visually pleasing or preferable to textual tools.
F, 4 of 5	Users quickly noticed differences between secured and hacked machines.
X, 2 of 5	Users would use Portall for directed investigation.
F, 2 of 5	Portall would speed up work practices.

Table 7: Major negative issues.

Type, # of users	Issue Reported or Comment Made
F, 4 of 5	Connection lines were confusing (too many or too difficult to trace); the display seemed cluttered.
F, 4 of 5	Marker for monitored host appearing on both client and server sides was confusing.
X, 4 of 5	Unfamiliar with Windows process names.
F, 3 of 5	Portall probably could not handle activity level of actual deployment.
I, 1 of 5	Need an activity baseline to be absolutely certain that any activity was suspicious.

Table 8: Key enhancements requested.

Type, # of users	Issue Reported or Comment Made
F, 4 of 5	Portall should let users filter out safe (or known) processes, IP addresses, protocols, and ports.
F, 3 of 5	Provide an indication of CPU usage by process.
F, 3 of 5	Let the user filter, minimize, or aggregate elements he has already examined.
F, 2 of 5	Allow users to move machine and process markers.
F, 2 of 5	Physically separate the markers for monitored machines from the unmonitored machines.
F, 1 of 5	Let the user set notification thresholds on packets/bytes/connections, etc.

The Expert group tended to report more Essential negative features, while the Moderate group reported more negative features that were due to their experience level. For enhancements and positive features, the expert group consistently generated more findings than the moderate group in every category except Experiential.

Portall's Key Strengths: All of my subjects were enthusiastic about my visualization in spite of its limitations. All said they would like to install a copy on their own machines, and all wanted to be included in future studies with more advanced prototypes. I found out later that several of the subjects had a reputation as "straight shooters" who would not have hesitated to "shred" my concept if they had not thought it was good. I was encouraged to hear that highly experienced users considered Portall a step in the right direction, and to know that they were not just being gracious when they made positive comments.

All of the subjects said that Portall's correlation of network traffic and process activity provided useful insight for their jobs. All agreed that the only alternative way they knew of to get this insight would be to mentally integrate the output of several other tools. Two users cited ways that Portall would speed up current work practices. One user said, "The only such 'tool' I would have [to do this correlation without Portall] would be extremely klunky, time-intensive attempts to do something similar manually."

All the users said they would use the visualization for informational monitoring, and only two said they would use the program for investigative analysis. This was unexpected since I designed Portall specifically for investigation. However, my study may have been biased toward an informational monitoring environment. The users might have responded differently if I first gave them reason to be suspicious via an IDS alert. Since the administrators I interviewed typically reviewed their logs once or twice daily, perhaps they expect to use Portall in the same ambient manner. Since most of the subjects only noticed suspicious events on their jobs once a week or less, I believe they are saying that they would use Portall more often than I had expected them to. Thus, I determined to minimize the amount of computational resources required by follow-on monitoring software so it could be used the way administrators desire.

Four of the five users quickly noticed important differences between the hacked and clean machines. Subjects noted large numbers of remote connections being made and dropped without any

user activity, suspicious process names, and a half-open covert channel to a keystroke-logging server process on the local machine. However, only two of them would commit to saying they positively saw something "suspicious" without an accepted baseline. Although I first showed the subjects the clean machine, one of the expert subjects did not accept that machine's behavior as a baseline for the other since he did not know the administration history of either host. I think a study with the participants using the visualization in their own environments would have helped determine the true effectiveness of Portall.

Finally, four of the five users made unsolicited comments about how they preferred the visual approach to text-based tools or how pleasing they found the visualization. (e.g., "I like the visual approach. [Portall] is much more visual than other tools.") One user enthusiastically said that if he had Portall in his office, he "would be watching it all day long!" I believe this is another indication that I am on the right track to give users what they need.

Portall's Key Weaknesses: Probably the most important conclusion we may draw from the negative issues is that my subjects, regardless of expertise, seem to prefer the visualization to represent machines as concrete physical locations rather than by their role in TCP communications. My target user population can be expected to understand the workings of TCP. Thus, I elected to base the visualization on TCP's client-server model. However, users did not understand why separate host markers for Dudette would appear on both the client and server sides (even though they clearly understood that the machine was running both client and server programs). Even the most expert user (who designs certification programs for a world-class computer security education organization) had to ask three times whether the separate markers for the monitored machine on the client and server sides were for the same machine. I conclude that my users expected to see each machine's marker appear in only one place.

Similarly, subjects did not readily grasp the meaning of the mapping I used to color connections and ports according to their TCP connection states. Although the users all understood TCP, they were not necessarily familiar with each state the protocol's connections could achieve. These results cause me to question whether a direct visual representation of TCP's abstract model was apt for the kinds of problems I was seeking to address.

Another problem users had with Portall was that they expected it to be a notification system rather than an analysis tool. Some users wanted the color red to be used solely to draw attention to potential dangers. Others wanted Portall to send e-mail or call pagers when certain threshold quantities were reached. One user suggested keeping a list of "known trojans" (malware) and highlight these programs if they appeared. Of course, this direction is contrary to Portall's purpose—to enable humans to effectively locate potential security compromises. There are plenty of fine notification systems in existence, and I do not wish to copy them. However, these comments indicate the potential for integrating notification-system data into Portall to expand its use for combined informative monitoring and investigative analysis.

Several of the users wanted to have more control over the layout of the icons. They suggested interactions such as manually dragging the icons and automatically organizing the host icons into several groups by the degree of confidence the user has that the hosts are benign and well-managed. All these suggestions indicate that users would like to be able to impose some ordering on the markers to help them make sense of the visual scene.

Three of the five subjects believed that Portall was not space-efficient enough to handle the activity levels they typically see at their jobs. One subject said that the one-Hertz data refresh rate was too slow and would miss important traffic in production use. To address the first issue in my final tool, I added

filtering and aggregation solutions proposed by the users. Ultimately, kernel modifications were needed to solve the refresh rate problem.

Key Enhancements Requested: A frequently asked question was, "Which one of these icons represents this machine (the one being monitored)?" The users suggested various layout, coloring, etc. schemes to distinguish monitored machines from the others. They also provided lots of ideas about filtering capabilities and additional indicators that they would need to perform their jobs. Users wanted to be able to minimize, hide, or aggregate known safe machines, communications, ports, and processes so that the picture only showed risky behaviors. They wanted to see indications of bandwidth, CPU utilization, and file-usage by each monitored process. One user stated that, "It's not necessarily the number of connections, but the amount of traffic going through the connections that I would look at, personally."

Some requests seemed to be leading toward making Portall an "intelligent" IDS tool, something it was not intended to be. However, a few enhancement requests that bordered on providing artificial intelligence were viable. These requests involved allowing the user to specify threshold values for bandwidth, CPU usage, and other quantities per machine, process, connection, or port. If a threshold amount was exceeded, the visualization would change the color of the marker in the display.

4.3.4.3 Portall Contributions

The Portall study showed that visually fusing host and network data was useful for investigating security-related anomalies. But the real host/network divide lies within the kernel of every major operating system today. My follow-on work[2] includes kernel modifications and compelling visualizations that free users from the restrictions imposed by today's tools and mindsets.

I made the following contributions via my Portall study:

- Designed, implemented, and tested a novel approach to security administration via visually correlating process and packet data.
- Demonstrated Portall's practical value by showing how users can detect malicious behavior resulting from real malware infections.
- Shown that Portall is useful for both informational security monitoring and for investigation analysis of anomalies.
- Collected detailed information about the kinds of filtering and aggregation users need to perform security monitoring and investigation tasks.

4.3.5 Key Conclusions

Portall illuminated the way to make further improvements in both my visualization design and my understanding of my users' needs as I built my ultimate implementation of tools and an infrastructure that would visually correlate host processes and network traffic data. Of particular importance was the lesson that modeling the TCP connection state with clients and servers as the protocol defines is quite confusing to users. I needed another way to present the same picture. Another key conclusion was that bridging the host/network divide with existing tools would not work. Netstat missed too much and tcpdump saw too much—together, the two made it appear that packets seen by tcpdump were actually connections missed by netstat. The result was confusion for the user but a lesson for me that showed how important it is to bridge the host/network divide.

4.4 The End of the Road

My evolutionary prototyping journey taught me much about the needs of my users and about what was technically feasible on a typical computer screen. Particularly, I learned that users are not always able to express what they want. When users told me they needed to see communication patterns and

large-scale visualizations of a network, I gave them the network view prototypes: the Network Pixel Map, Network Eye GL, and VISUAL. The users' reactions showed me that these representations would not scale up to my users' normal environments. Their comments also told me that what was needed was a way to relate network events to the host processes that caused them to occur. This required a host process visualization and a way to link it to the network visualization. Creating this linkage uncovered the existence and the extent of the host/network divide. As I tried various ways to bridge this divide, I discovered that kernel modifications were the only way to go. The next chapter discusses the tool and infrastructure that actually bridges the divide, HoNe.

CHAPTER 5 THE SOLUTION

5.1 Reasons for Creating HoNe

During interviews and usability evaluations of prior tools, I had greatly refined my understanding of system administrators' security visualization needs. In light of this earlier research, I designed HoNe to remedy the address as many of these needs as possible. I made improvements over earlier prototypes in usability, function, and architecture.

5.1.1 Usability Improvements

One of the greatest drawbacks to Portall's approach was how I chose to model the interface after the TCP/IP client-server model. Since most machines function as both client and server, two icons appeared for each machine—one on the client side and the other on the server side. This feature confused even expert users. In contrast, HoNe keeps a single representation of a machine in one place according to its trust category rather than splitting up a machine into its client and server components.

Another shortcoming of Portall was that it could only display 30 or fewer hosts on a side. HoNe has a much greater spatial capacity than Portall because it has superior filtering and is able to magnify or shrink entire regions independently. Thus, many more connections and hosts may be shown in HoNe.

While Portall connected processes to their host using lines that were easily mistaken for communication lines. In contrast, HoNe's process icons partly overlaps the owning host's machine icon. This *containment* placement approach was much clearer to users. Containment does a better job at visually separating processes running on separate hosts while visually joining processes with the host they run on.

One important visualization feature that Portall was missing was the ability to allow users to define the positions, colors, and other preattentive features of boxes and communication lines. HoNe is designed to support this highly desirable feature-definition capability from the start.

5.1.2 Functional Improvements

When asked what level of temporal detail was needed to perform security analysis, my users told me they needed to distinguish events at the microsecond level. When I asked them over how long a period of time they might conduct an investigation, they told me they might look back 60 days or more. Using a multi-level overview plus a detailed view, HoNe can represent multiple orders of magnitude in the time scales from several months of data down to the microsecond level.

HoNe uses an embedded database to allow detailed queries with the full power of SQL. Such queries were not possible with Portall, which kept its information in a database but did not restructure the data to support querying. The embedded database allows users to construct filters for visual elements based on whether the object filtered for matched the query or not. Embedded SQL gave HoNe users the ability to find and highlight all connections no matter how complex the filter statements. These filters provide the basis of the user-defined coloring, etc. and users found them quite effective to quickly identify problems.

5.1.3 Architectural Improvements

Probably the most significant departure of HoNe from Portall is that HoNe accurately traces packets to the processes responsible for them rather than using an approximate method of trying to correlate packets to sockets seen in netstat data. Portall could not distinguish process events that occurred within the same netstat polling period while HoNe does not miss any socket activity at all.

HoNe finds the process responsible for each packet that comes into the machine. This feature is discussed in greater depth in section 5.4.2.

While Portall was written in C# and relied heavily on the Microsoft Windows .NET framework, the HoNe Visualizer is implemented in Tcl/Tk and is platform independent. This was important for my users, most of whose important servers were running on Unix-like operating systems. I separated the Visualizer from the HoNe correlation engine so that a user could view data from multiple heterogeneous machines on a single workstation.

5.2 Description of a successful hack

In this section, I will introduce HoNe by showing what happened on a monitored machine when it was hacked. This was a real incident, not a simulation. Figure 40 shows the state of the machine during the critical first 20 minutes of the intrusion.

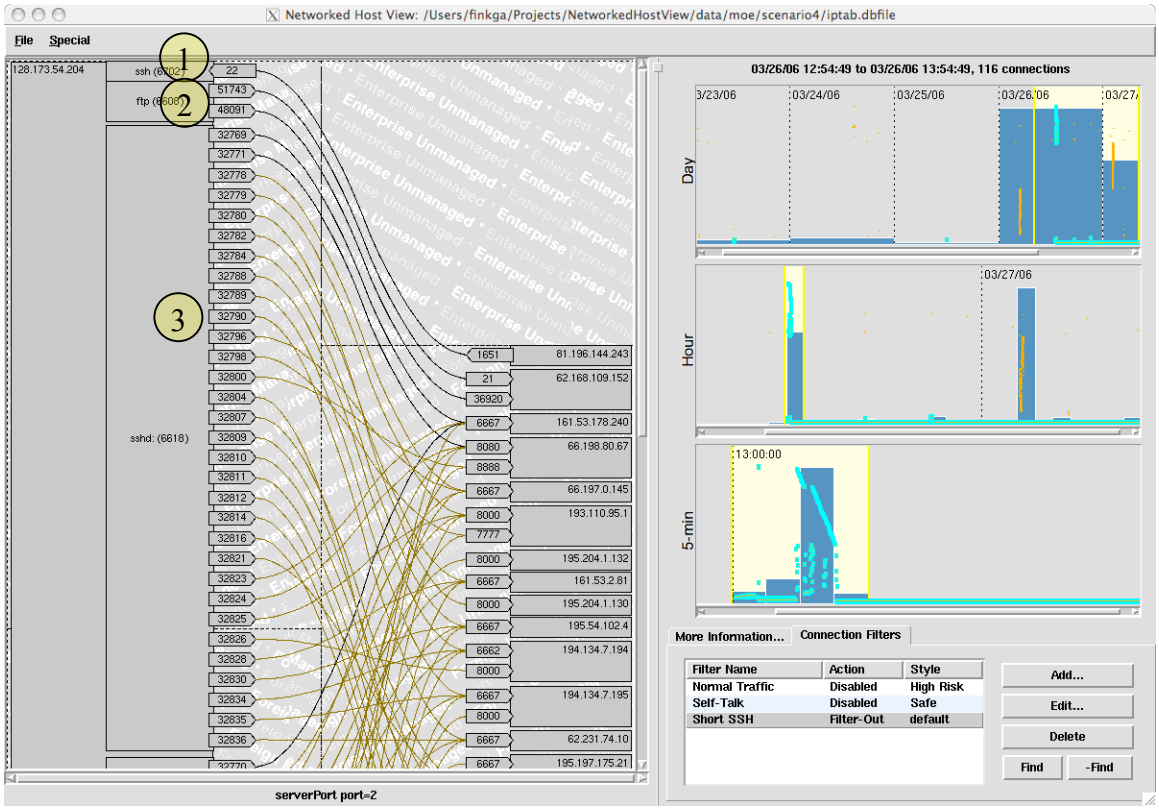


Figure 40: Snapshot of a successful hack: (1) Login, (2) downloading a toolkit, (3) starting the IRC 'bot.

In Figure 40 the user has opened a connection database for monitored host 128.173.54.204 and focused on the area of interest using the Day, Hour, and 5-minute overview histograms on the right. On the left is the detailed view of what happened in the critical first minutes of the hack. The events are roughly in time order from top to bottom, so we can trace the progression. First the intruder logged in from remote host 81.196.144.243 using a password he had broken earlier. We know he already had the password because there is only one login attempt. Next he started ftp to download files (probably his toolkit) from a remote machine. Finally, he started up his exploit program (evidently an Internet Relay Chat (IRC) 'bot) to allow other users to make connections to this machine. Using the IRC bot,

the intruder later attempted to gain root control of the machine and use it to attack other machines. However, at that point the machine crashed and I stopped the experiment.

5.3 The HoNe Visualizer

This section presents the HoNe Visualizer's user interface and explains its operation. The main window layout shown in Figure 41 is divided into two halves: the left half is the detailed view (numbers 1 to 4), and the right is the control pane (numbers 5-8). The detailed view is subdivided into four trust regions: 1) *Enterprise Managed*, 2) *Enterprise Unmanaged*, 3) *Foreign Unmanaged*, and 4) *Foreign Managed*. Machine icons appear on the left (*Enterprise*) side when they belong to the user's company, otherwise they are considered *Foreign*. Machines that are administered and monitored by the user are considered *Managed* and appear in the upper regions. A machine should be considered *managed* if it is running the kernel module and is providing reports to the user. Other machines are considered *Unmanaged*, and appear in the lower regions. In the figure, there is only one enterprise-managed host, 128.173.54.204. The other hosts are foreign-unmanaged machines. The user is responsible for entering the IP addresses and ranges of machines used for this classification into the configuration files prior to running the visualization.

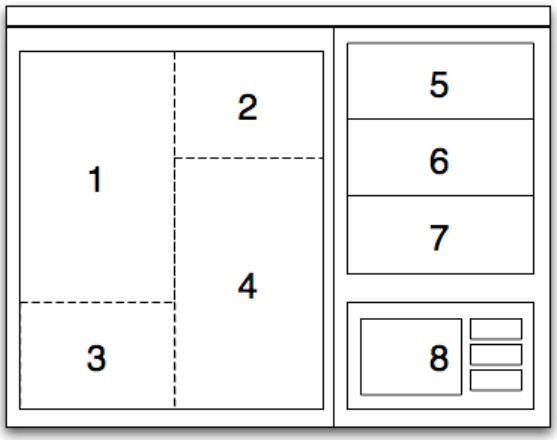


Figure 41: General layout of the HoNe main window

The right-hand pane shown in Figure 41 contains three histogram windows at different time scales, 5) a daily view, 6) an hourly view, and 7) a five minute view. These overviews show the traffic levels and connections using a histogram for which each bar represents a day, hour, or five-minute period of time. Item 8 is a tabbed pane with the more information window in one tab and a list of SQL filters in the other. All these items will be explained in later paragraphs.

In the following description of the visualizer's user interface, I will refer to another intrusion incident I captured using HoNe. Figure 42 shows an internal system (128.173.54.204, the virtual machine) as it is being hacked by several cooperating external machines. The status line at the top on the right side indicates the events shown occurred on 12 Jan 2006 from 02:43:50 to 02:57:30 in the morning. Note: These events appear in the detailed view slightly out of time sequence looking top to bottom. This is because a connection to one machine on the left occurs in the middle of several connections with another machine. I have elected to keep the machine icons together rather than to keep the time sequence intact.

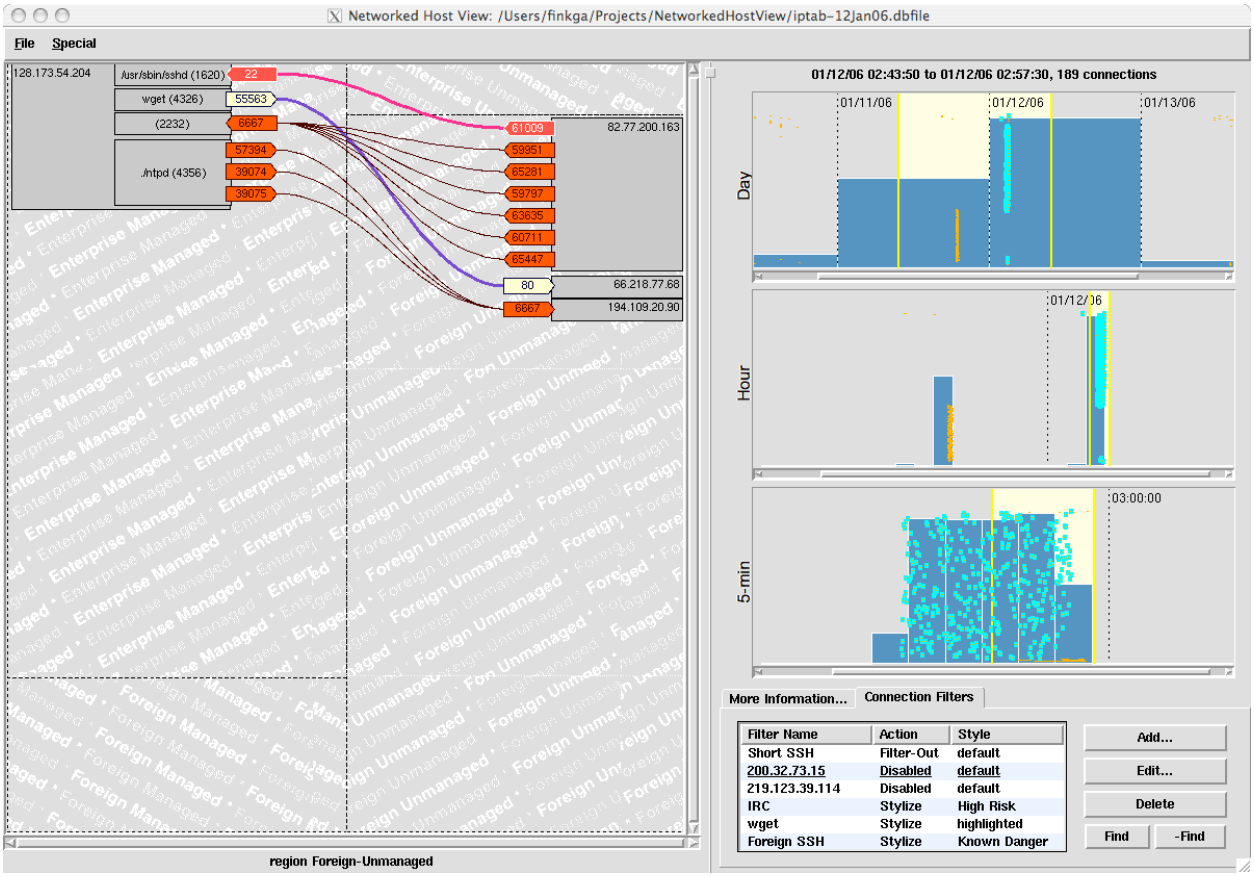


Figure 42: A screenshot of HoNe's visualization. The visualization displays connection data from an SQL database generated by the modified Linux kernel to visually correlate each packet to a running process.

First we see a barrage of SSH login attempts (highlighted in cyan on the time window panes on the right) from foreign host 200.32.73.15. This barrage started at about 02:31 and continued for 23 minutes. There had been similar barrages during the previous evening and four days prior. Because this SSH barrage continues after the actual hack has started and from a different IP address than the attacker who gets in, I assume it is an attempt at covering the actual intrusion. The attacker has already guessed the password, probably from an earlier SSH login attack. The intrusion happens when the attacker logs in from 82.77.200.63. Next we see the attacker using wget to download a toolkit from 66.218.77.68, and a few seconds later, the machine is hacked, having opened up an IRC server (unnamed process 2232) and client (innocuously named "ntpd" but making foreign connections to port 6667. Because the entire intrusion took place within 3 minutes, including downloading and starting the IRC bot, I believe this attack was automated. Full forensic analysis was not required for this incident, so this is not a definitive analysis.

The detailed display (enlarged in Figure 43) shows each host by IP address (and DNS name if known). In Figure 43, the host icons are items (1) AND (6). Item (1) is the monitored host and it is in the enterprise-managed zone. The host icons pointed to by item (6) are foreign-unmanaged machines. For managed hosts, HoNe shows the processes involved in the communications displayed inside the host box (icons pointed to by item (2) in Figure 43). Processes contain the executable name and PID if known.

Bristling from the inside edges of the processes or machines are the ports the machines are using to communicate (pointed to by items (3) and (5) in Figure 43). Ports are shaped like arrows with listening (server) ports pointing toward their machine and initiating (client) ports pointing away. Item (3) in Figure 43 is a server port attached to the SSH service on the monitored host. Item (5) is a client port used by some unknown process on the foreign-unmanaged host, 82.77.200.163. The arrow shapes help users to clearly see who initiated each communication.

Communication lines (as pointed to by item (4) of Figure 43) join the client and server ports. Item (4) in Figure 43 points to a connection between an ephemeral port on the client side an the managed host's SSH server port on the server side of the connection. The direction of the arrow determines which side is the client and which is the server. Communication lines and port, process, and host icons may be manually colored by the user or they may be colored according to some filtering expression provided by the user. In this case, the user has elected to highlight any successful incoming SSH sessions initiated on the monitored machine by a host in the foreign-unmanaged zone with the "Known Danger" color scheme (white text on a bright red background).

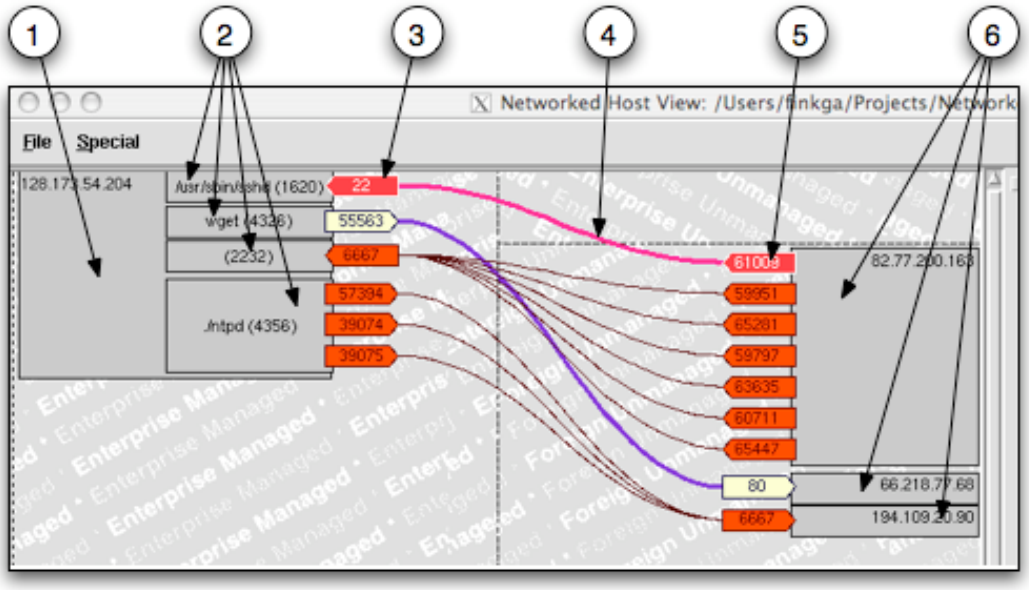


Figure 43: An enlargement of the detailed view of the HoNe visualizer.

HoNe provides multiple levels of detail so that the user will not have to look at too many items at a time in the detailed view. However, the nature of Internet activity makes it possible for thousands of activities to happen within a fraction of a second. Thus I have included a way for users to zoom or shrink a region to fit it into the detailed view as shown in Figure 44. In this example, nearly 100 foreign-unmanaged machines are making connections to the SSH port of the monitored machine, but the foreign-unmanaged region has been shrunken so that they fit into a fraction of the screen. At the same time, the enterprise-managed region has been enlarged.

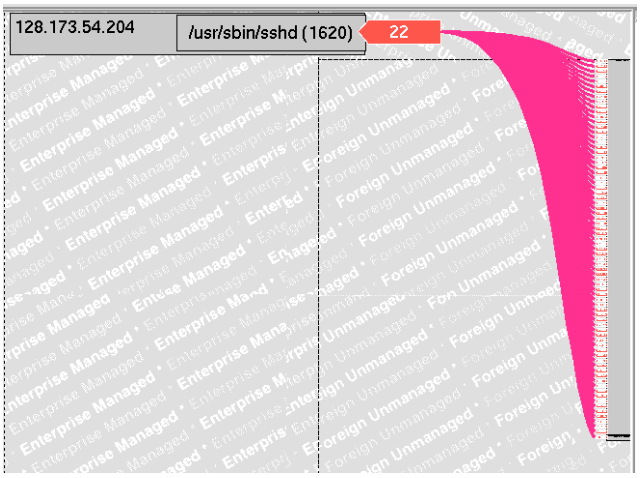


Figure 44: This figure (shown at the same scale as Figure 42) shows how regions may be magnified or reduced independently. Here, the Enterprise-Managed region is magnified several times while the Foreign-Unmanaged region has been reduced to show all of the numerous SSH connections.

The right pane of the main window primarily contains controls that determine what the detailed view shows. The top section shows three histogram timelines that form the connection overview (see the close-up in Figure 45). The bottom section is a tabbed area where the Connection Filters and "More Information" panes reside.

The connection overview (Figure 45) shows the entire database file at a glance on three separate scales. The reason for separate scales is that users stated that they needed to distinguish events at the microsecond level, but when asked how much data they might look at to investigate intrusions, they said they might want two or more months' worth. These viewing levels cover about 13 orders of magnitude (from 10^{-6} to 10^{7} sec). The tri-level overview plus the detailed view is the way I chose to span this large shift in magnitude.

The overview was inspired by histogram sliders[69], a new widget designed to give instant feedback to help the user locate where on the slider scale the most relevant information lies. The timelines represent the passage of time from left to right with earlier events placed closer to the left. The relative number of connections within the time period the bar represents determines the height of each histogram bar. I multiply height in pixels of the timeline by the number of connections within the time period of the bar divided by the total number of connections displayed in the whole histogram to derive line height of each bar. So the top histogram in Figure 45 has one bar that contains most of the connections for the whole four day period. Placing the mouse over a histogram bar shows a "tool tip" window with start time of the bar and the number of connections within its duration.

The pale yellow area bounded on the left and right by bright yellow bars is the area of interest for each timeline. Users can move the whole area of interest or slide just the left or right bars. The width of the area of interest is a focal area that determines the amount of time that will be displayed in the next lower view.

The topmost timeline's histogram bars each represent one day's worth of data. Fifty or more days' worth of data can be shown here concisely. When the user selects an area of interest in this histogram (of no more than 1.5 days in duration), HoNe displays that area in the second histogram, where each

bar stands for an hour of data. Similarly, selecting an area of interest (no longer than 6 hours in duration) here brings up the bottom histogram with a bar for each five minutes of data. When the user selects an area of interest in this lowest histogram, the detailed view pane shows the hosts, processes, ports, and communication lines in that time window. The information line above the histograms tells the extent of the finest-granularity area of interest that the user has selected and the number of connections it contains.

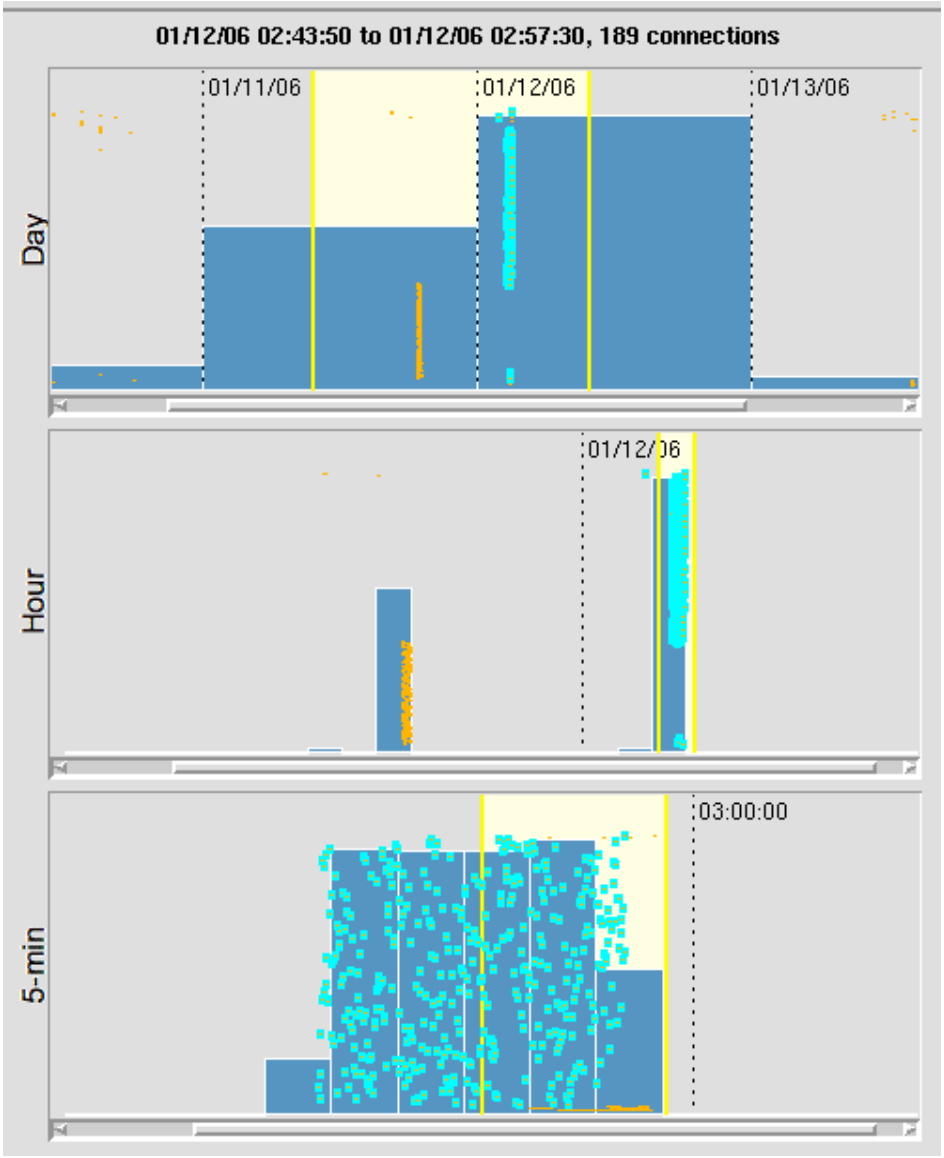


Figure 45: Detail of the connection overview histograms. Connection lines are orange but highlighted in cyan when selected. Selecting any connection highlights it in all views.

The histograms are overlaid with horizontal orange lines that represent individual connections. The horizontal length of the connection line represents the duration of the connection. Because the scale may force some connection lines to zero length, I have constrained all lines to at least 2 pixels length. The longer duration a connection has, the nearer it is placed nearer to the bottom of the histogram. The metaphor I have used is "longer connections are heavier and they sink to the bottom." Placing the

mouse over a connection line shows a "tool tip" with the source and destination IP addresses and ports and the measured connection duration in seconds.

When a user selects a connection or executes a search, the selected or matching connections are highlighted with a cyan border. In Figure 45, the user has executed a search to highlight all SSH connections initiated by foreign-unmanaged hosts to any managed host. The matching connections are highlighted in every view (detailed and all three histograms) where they appear.

The bottom section of the controls pane has tabbed windows with "More Information" (Figure 46) and Connection Filters (Figure 47) tabs. The "More Information" tab is not a control but a message box containing detailed information on the current selection. When users double-click connection lines, all the packets associated with that connection are displayed in this window. Double-clicking a host icon spawns a "whois" command to find information on the host's IP address. An example of this output appears in Figure 46.

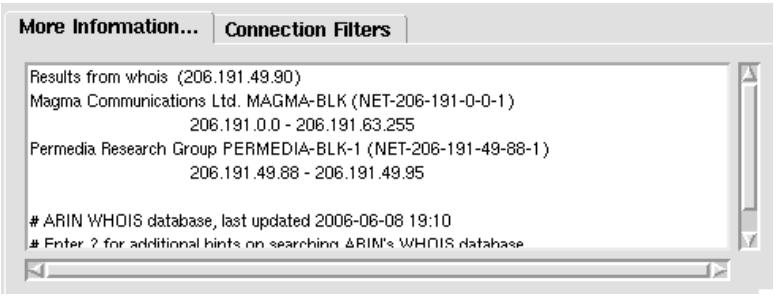


Figure 46: The More Information tab.

Double-clicking on a port retrieves the information from the machine's /etc/services file that tells what protocols have registered the use of that port. In the future, this action might retrieve current information about what malicious programs are known to use that port number to communicate. Another future expansion of the "more information" tab would be to show what files a communicating process had open over its lifetime when the user double-clicks its icon. My users expressly requested each of these "more information" features during my pilot study.

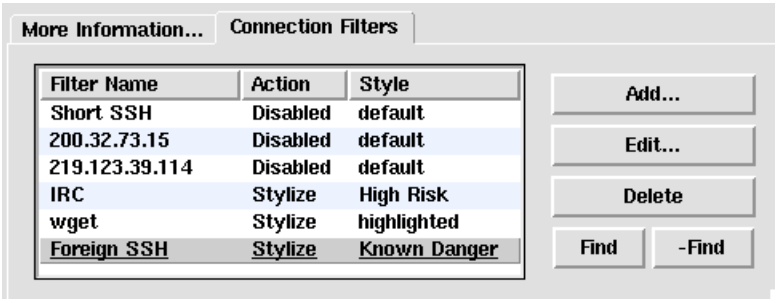


Figure 47: The Connection Filters tab.

The second tab of this section contains connection filters (Figure 47). Filters are SQL expressions the user can employ to remove extraneous display items from the detailed view and focus on what is important. Filters may have other actions besides removing items from display. Filters may omit or stylize any display item (host, process, port, or connection) in the detailed view using the full matching power of SQL-92 queries. Additionally, one can use "filter-in" filters to display any items that would have been removed by a "filter-out" filter. This lets the user filter out, for instance, all Secure Shell traffic broadly, and then perhaps filter back in traffic that comes from a particular range of IP addresses.

Users may also create a filter from something that is already displayed using a query-by-example technique.

An important use of filters is to highlight in all views the connections that either match or do not match the filter's SQL. This capability allows users to quickly find features that match a search criterion. When the user selects a filter and presses the "Find" button, all connections that match the filter are highlighted in any views they appear in. Conversely, pressing the "-Find" button performs an inverse find, highlighting all the connections that do not match the SQL expression.

In practice, I found that users who employed the "Find" and "-Find" capabilities were able to hone in on an area of interest fastest. For instance, if a user believed that long-duration SSH connections were an indicator of a break-in, he might make a filter that contained the expression:

```
svPort = 22 AND (lastSeen – firstSeen) > 20
```

This expression would match all connections where the server port number was 22 (SSH) and the connection duration was greater than 20 seconds. The user could then find all connections that matched this criterion and zoom in on times where these kind of connections occur.

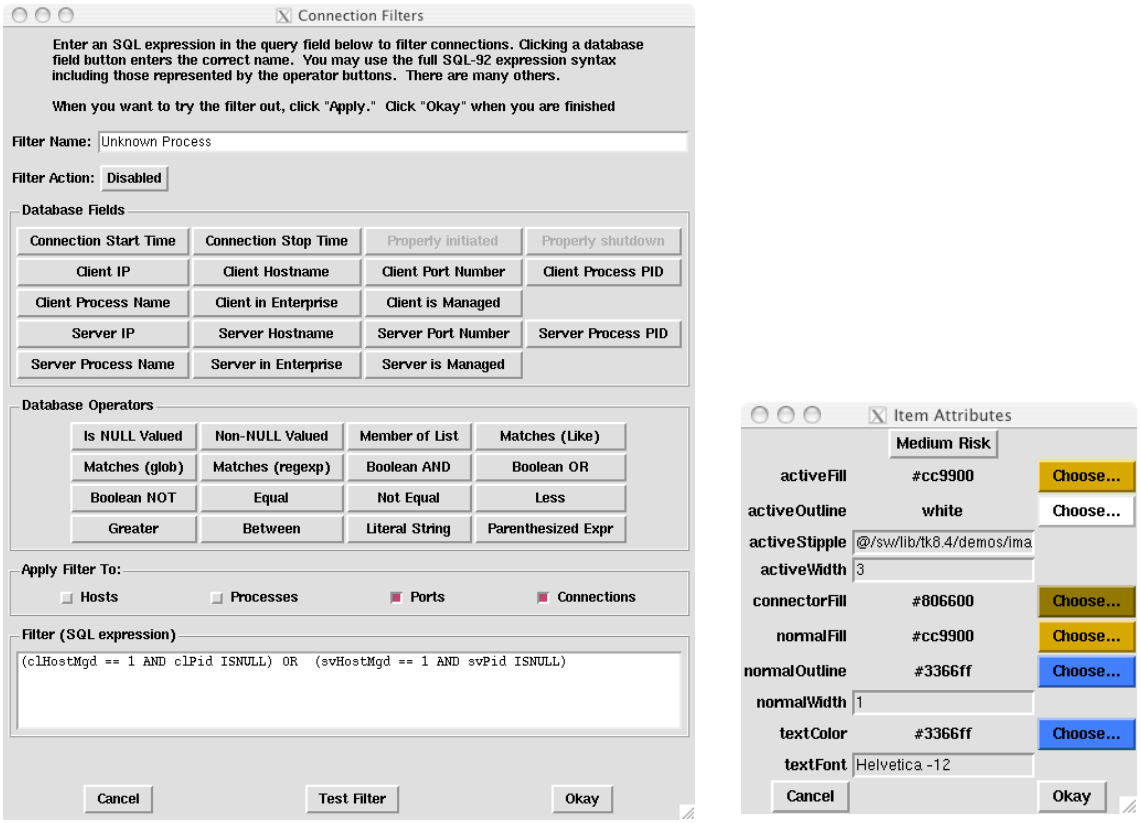
Users may define arbitrarily complex filters, but the best practice indicates using a series of simple filters in succession. Filters are applied from top to bottom with later filters acting on the output of former filters. Thus a user may filter out all short SSH connections and then color all remaining SSH connections red with a separate filter. After filtering out whole classes of connections, a user may use a subsequent filter to restore a subset of the previously removed connections that match certain criteria.

Filters may be defined using a dialog box shown in Figure 48, part (a). A filter may change the display of any matching item to any of eight predefined styles: Default, Highlighted, Safe, Low-Risk, Medium-Risk, High-Risk, Known Danger, and Unknown (see Figure 48, part (b)). The user may change colors, fonts, and line thicknesses for each of the predefined styles as desired. Additionally, users can style individual items manually, apart from filters, to mark items of interest. The intention is for users to highlight items of special interest with visual characteristics that are meaningful to them. I found the combination of powerful data filters and user-definable visual styles very useful for analysis.

5.4 HoNe Infrastructure

HoNe is more than just a visualization, it has a supporting infrastructure that enables true packet-process correlation. HoNe visualizer, together with two or more monitored hosts running the HoNe infrastructure, can enable users to see end-to-end across the network. This section presents the implementation of HoNe's infrastructure, the bridge processes, and the database.

HoNe's infrastructure creates the packet-process correlated data used by the visualization but is completely separate from it. In this section I present the design of the kernel modifications, the bridge processes, and the database that support the visualization.



(a) (b)

Figure 48: (a) The HoNe filter definition window, and (b) the style definition window.

Figure 49 is a block diagram of HoNe's architecture showing where in the networking stack each of HoNe's components is located. Figure 50 is an informal dataflow diagram of HoNe showing the data and control flows between HoNe components. In these two figures, the loadable kernel module (LKM) portion of HoNe resides in the network and interface layers and responds to every packet entering or leaving a network interface. The LKM hooks into the Linux Netfilter[4] framework and may be loaded or unloaded by a privileged user process. If the kernel modifications could be confined to the LKM, they would be much more portable, but the kernel's transport layer (TCP and UDP) must also be modified to allow tracing packets to their destination processes. The HoNe visualizer will eventually allow the user to load and unload the LKM, turning packet-process correlation on and off. The LKM logs each packet and its associated process to a text file called the connection log file. The bridge process reads the connection logs and translates them into connection databases. Log files may be from any monitored host running the kernel modifications. By joining logs from more than one monitored host, HoNe may obtain an end-to-end view of process communications activity. In an operational environment, the database translation should be controlled by the HoNe visualizer, but this coupling has not been done yet. The HoNe visualizer reads connections from various connection databases and presents the results to the user in graphical form.

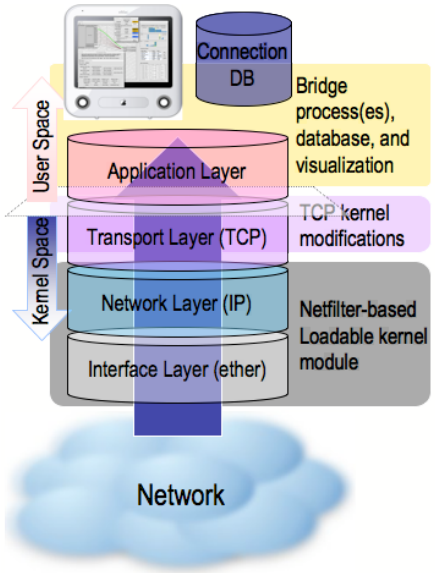


Figure 49: HoNe architecture block diagram.

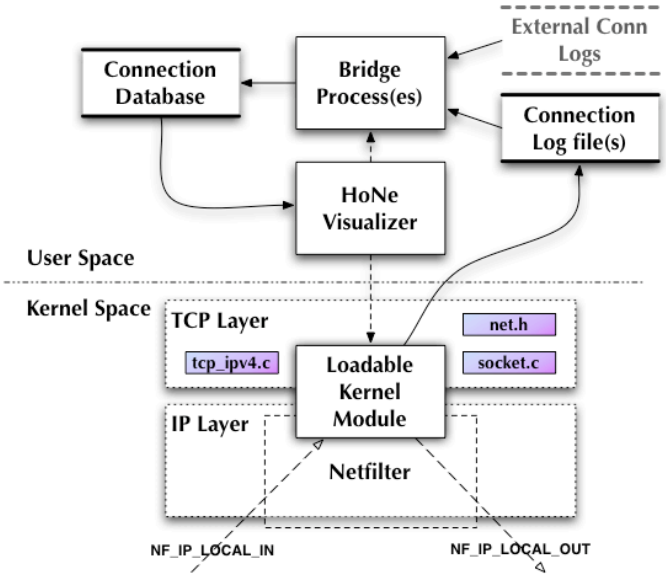


Figure 50: HoNe data and control flows.

5.4.1 Justification for My Approach

Some investigation into why no packet-process correlating software existed in the literature or practice revealed that the separation between host and network was enshrined in the networking stack inside the kernel. The problem, as illustrated in Figure 51, is that given a modern operating system kernel and network stack based on a layered networking model[70][71], we cannot at the same time know both the source and destination machines and the associated processes for a given incoming packet.

This is not an oversight in the kernel design so much as it is an intentional separation of concerns characteristic of the layered approach. Modern networking models divide the work of a communication protocol into separate layers where each layer has the property that it only uses the functions of the layer below, and only exports functionality to the layer above. The set of layers is called the *protocol stack*. The logical separation makes the protocol easier to understand and allows separate vendors to provide different layers because they must agree only about the layer interfaces.

However, the layered model implies that the transport layer (TCP/UDP) may only call functions of the network layer (IP), and IP may only provide functions to TCP/UDP. To correlate packets to processes, TCP would need to call a function in IP that returned the source and destination address, but no such function exists in the standard because routing information is irrelevant to the transport layer. Another possibility would be for IP to call a function from TCP to find out what process the packet is destined for, but this directly violates the layering principle. Thus the correlation I need to do is impossible given faithful renderings of the standard protocol stacks.

For incoming packets, (number (1) in Figure 51) we know the source host at the IP layer only. When the packet reaches the transport layer (as in arrow (3) in Figure 51), this information is no longer available. In the application layer there is no problem for outgoing packets (number (2) in Figure 51) since the destination host is known, but once a packet reaches the network layer, its source socket (and thus process) is no longer known. So the correlation engine could not be confined to any single layer in

the protocol stack. This inconvenient fact implied modifying the kernel itself, something I was loath to do. The following are the implementation alternatives I tried:

- Integrating process data from tools like netstat and lsof with packet data from tcpdump into a single picture
- Using firewall redirection rules and divert sockets to copy all network data to an accounting process
- Modifying GlibC to intercept all *socket()* calls and tracking the processes that made the calls
- Modifying the kernel itself

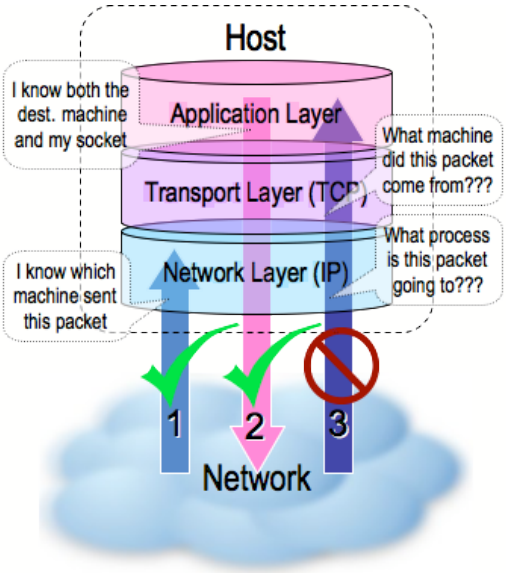


Figure 51: In network stacks based on the ISO layered networking model, we cannot simultaneously know both the source host and destination process for incoming packets.

Integrating netstat and lsof with tcpdump: The first solution is very complete except that netstat and lsof rely on timed polling and can only provide snapshots of the machine's state. Thus, extremely rapid events like infection by the SQL-Slammer worm can easily slip by unnoticed. By the time tcpdump had captured a packet, the socket that created or received it would be gone and with it any hope of connecting the packet to a process. Based on my attempts to detect the nmap port scanner running on the monitored machine via netstat and lsof, it appears these tools cannot track incomplete or very brief connections. Thus, most scanning behavior (inbound as well as outbound) would be missed. Outgoing packets that were rejected by the firewall also never created an entry in netstat's list of sockets, implying that attempted communications originating from some unauthorized process on the monitored host (a very important indication of intrusion) would be missed. Finally, this approach is unsatisfactory because the host and network data must be retrieved separately and thus have separate timestamps. Any packets that cannot be matched to a running process are suspicious, and if the data collection is not truly integrated, mismatches due to the separate data collection will produce false positives.

The standard netstat can query network data structures as often as once per second. While this seems very frequent, it is important to realize that many connection activities are happening every

microsecond. To see whether or not netstat could capture all the information needed, I wrote a simple Tcl script that retrieved random web pages at normally distributed random times with a mean of 5 seconds and a standard deviation of 1.5 seconds. I collected and examined the netstat data from the time while the tool was running and found that whole connections were missing even when netstat was running once per second. So I modified the netstat source code removing the sleep interval to see how well netstat could do if it was constantly polling the kernel without sleep. The resulting program dominated the processor and generated 22MB of text per minute when run against the web surfing script. The output had very little entropy, evident from the fact that it could be compressed by greater than 99%. Even with this extremely fast collection rate, netstat still missed parts of the web conversations and important socket state transitions. Thus I determined that netstat was insufficiently accurate even for low data rates no matter how frequently it collected data.

In contrast, tcpdump catches all packets received by the machine (unless the kernel drops them because of overload). So it would be theoretically possible to use tcpdump packet traces to recreate conversations that netstat was missing. However, consider that an attacker could use this knowledge to plant fake (spoofed) conversations on the network to make it look like a connection was in a different state than it truly was. Then when a packet flow that did not match any connection seen by netstat or lsof, the tool would have to choose between believing tcpdump (subject to spoofing) or netstat (subject to missing events). The netstat/lsof/tcpdump solution is incomplete and too loosely integrated to be useful.

Firewall Redirection: To ensure that I accounted for all the packets while the sockets were still active, I needed to move closer to the kernel. By putting special packet redirection rules into the monitored host's firewall, I could catch all packets going into and out of the machine and trace them to the related socket before it disappeared. The firewall's rule would copy all incoming and outgoing traffic to a special "divert socket" owned by an accounting process that would correlate each packet to a running process or determine that no such process exists. Figure 52 illustrates this approach.

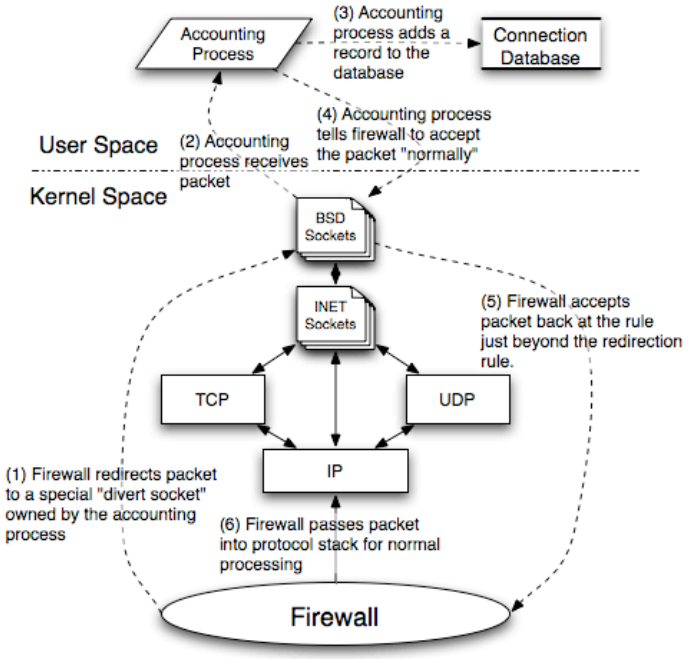


Figure 52: Correlating processes to network activity via a special firewall rule.

A shortcoming of this approach is that using current firewalls with packet-diversion capability, (e.g., NetBSD's ipfw and Linux's iptables) one must reinsert the diverted packet back onto the machine's networking stack for processing (step 5 of Figure 52). Reinserting the packet at the proper place in the firewall's rule set proved to be tricky to do without changing the effective behavior of the firewall. This method also contacts the accounting process for every inbound packet that enters a monitored interface regardless of whether the machine was listening on that port. Thus, the accounting process becomes a performance bottleneck and makes the monitored machine much more vulnerable to denial of service. Any instability in this process could tie up the firewall and crash the machine making the firewall-divert-socket approach inherently unstable.

Modifying Socket Libraries: The third approach would be to modify the program libraries responsible for creating and maintaining sockets (particularly GlibC) so that I could maintain a list of all open sockets and the programs responsible for them. This approach would work for all executables that were statically bound to the socket libraries, but it would not be able to track the activities of processes whose executables were dynamically bound, nor would it prevent programs from avoiding our modified library by using one of their own. In my usability study with Portall, I found that bypassing these libraries was a common tactic used by malware authors to avoid detection.

Kernel Modifications: Because none of the other methods worked, I was constrained to modifying the kernel itself to make the required packet-process correlation. The modified kernel intercepts and tracks each packet that belongs to a running process. This approach has the advantage of showing only the data that actually had an effect on the monitored host. "Script kiddie" attacks and other noise that doesn't affect a process on the machine simply don't appear. I chose an implementation that had the smallest performance impact on the kernel. I simply log the header of every packet associated with a socket and the name of the process that created the socket to a text file. I then process the text file into an SQL database and visualize the data via a Tcl/Tk user interface. The performance impact to the kernel is no worse than running tcpdump, and there is no impact when the kernel module is not loaded.

An alternative to the modified kernel would be an *instrumented* kernel, where certain actions such as receiving a packet triggered handler routines. These handler routines could be used to do the packet-process correlation. Sun's DTrace[33] is capable of such kernel instrumentation without modifying the kernel at all. At the time I created the bridging architecture, I was unaware of DTrace, but using it instead of custom kernel modifications could be an excellent direction for future research.

5.4.2 Kernel Modifications

There are many ways one might modify the Linux kernel to obtain the correlation needed by a true networked host view, but Figure 53 through Figure 55 show three possibilities. In the first possibility, the forward method, the LKM retrieves the packet in the network layer and looks forward (from the point of view of an incoming packet) to the transport layer to find out what TCP or UDP socket (and thus what owning process) is associated with the packet. While all the possibilities break traditional layer boundaries, the forward method is the simplest to implement and requires the fewest modifications to the kernel outside the LKM. However, this method causes TCP to do its socket-lookup work an additional time, and even forces a socket lookup for retransmitted packets and other errors that TCP would not normally be required to work with.

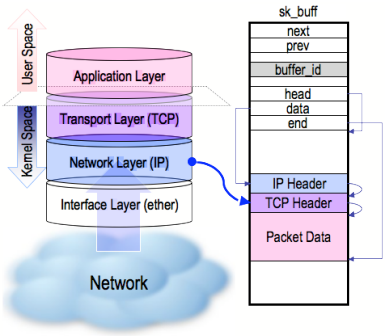


Figure 53: The Forward Method.

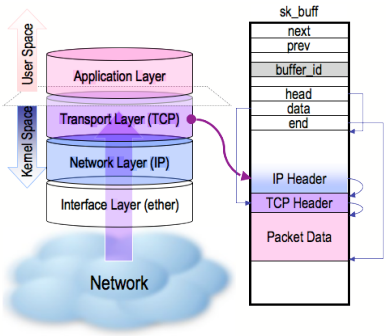


Figure 54: The Reverse Method.

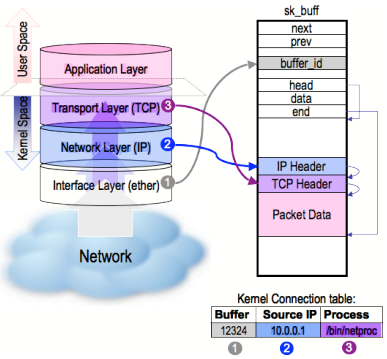


Figure 55: The Progressive-Forward Method.

Figure 54 shows the reverse method where packets are correlated in the TCP layer instead by looking backward (from the incoming packet's perspective) into the network layer for the source and destination host IP addresses. This method is more efficient than the forward method but more intrusive. Netfilter provides visibility into the network and datalink layers only. Because my solution is an LKM that uses Netfilter's hooks, this method implies changes to the transport layer beyond what can be done within a kernel module. Another problem with the reverse method is that because the IP header may be variable length, HoNe cannot always be sure of where the header starts, and worse, especially for outbound traffic, we have no guarantee that this header contains fresh data.

Figure 55 shows the progressive forward method where we may assemble the packet-process correlation as the packet traverses the stack. When the sk_buff data structure (the buffer containing the raw packet and other information) is created, the kernel would assign it an identifier and store this as the primary key in a socket record. In the network layer, the kernel module records the source (for incoming packets) or destination (for outgoing packets) IP address. In the transport layer, the kernel records the owning process name by tracing the socket that the packet belongs to. When all three pieces are assembled, the kernel writes a record to the connection log. This method is much more accurate than the others, and it has the advantage that if there is an error and the packet does not go to both layers, no erroneous connection log entries will be generated. However, this would require the equivalent of a simple database to be resident within the kernel—an enormous amount of overhead in a place where every added line of code is heavily scrutinized.

I chose the first method, the forward method, as the means of packet-process correlation. It is the simplest, least intrusive method and it works adequately. The progressive forward method would be superior, but I believe it is not a realistic option at present.

5.4.3 Bridge Processes

The LKM writes out the connection log as a text file, but the visualization requires a database to support queries. I needed a bridge process to take entries from the text file and place them in the database. Every line of the text file contains the following information:

- Time stamp of when the process was recorded
- Process identifier (PID) and name
- Connection direction (relative to the monitored host)
- Socket state

- Length of the packet
- Binary dump of the IP and TCP packet headers

The information in the packet header is used to reassemble TCP conversations (so that all packets from a single connection can be grouped together) and to create a text packet dump similar to what tcpdump generates. Figure 56 shows the information that is contained in these packet headers.

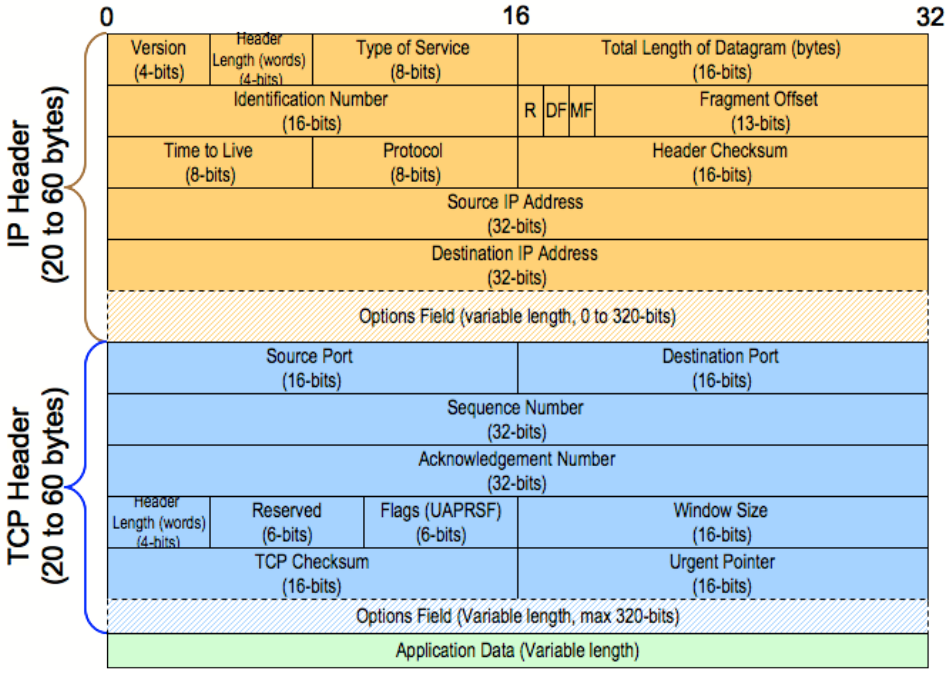


Figure 56: IP and TCP packet headers.

After gathering this information, the bridge process inserts records into the database for the host, process, socket, connection, and packet database entities involved with this communication. Before I implemented packet-process correlation in the kernel, I used netstat and tcpdump data to accomplish a loose approximation of the correlated data. I created two other bridge processes, one for translating netstat data and another for translating tcpdump data. These processes work in tandem to populate the database. When tcpdump and netstat data are combined in this way, the result contains a superset of the truly correlated data with many extraneous "connection" objects that come from tcpdump packets seen on the wire that do not actually impact the host. However, the ability to accept data from netstat and tcpdump was very useful in designing the usability evaluation and may also be useful for certain user tasks.

5.4.4 Database

This section describes the database that HoNe uses to store connection information in a form that may be queried. HoNe uses an embedded database called SqlLite3[72]. I selected this database because it was faster and less complex to set up than a traditional database management system, it has bindings to Tcl (the scripting language I chose to implement HoNe Visualizer in), and it is free. Sqlite provides the complete SQL-92 query syntax lending a great deal of power to the application.

5.4.4.1 Entity/Relationship Data Model

We can understand the database design better by looking at an Entity/Relationship Diagram (ERD) of the data to be stored. From the diagram, I will derive the database structure. The entity sets modeled are:

1. *Hosts*: Machines whose primary identifier is an IP address.
2. *Processes*: Programs running on monitored hosts.
3. *Sockets*: A unique combination of a host IP and port number that serves as a communication endpoint.
4. *Connections*: A unique socket pair.
5. *Packets*: Chunks of information that pass across a connection as units in communication streams.

The relationships modeled are:

a. Every *connection* must have exactly one client *socket*.
b. Every *connection* must have exactly one server *socket*.
c. Every *packet* may belong to up to one *connection*.
d. Every *socket* belongs to exactly one *process*.
e. Every *process* is local to exactly one *host*.
f. Every *socket* belongs to exactly one *host*.

These entities and relationships are illustrated in Figure 57 and discussed below.

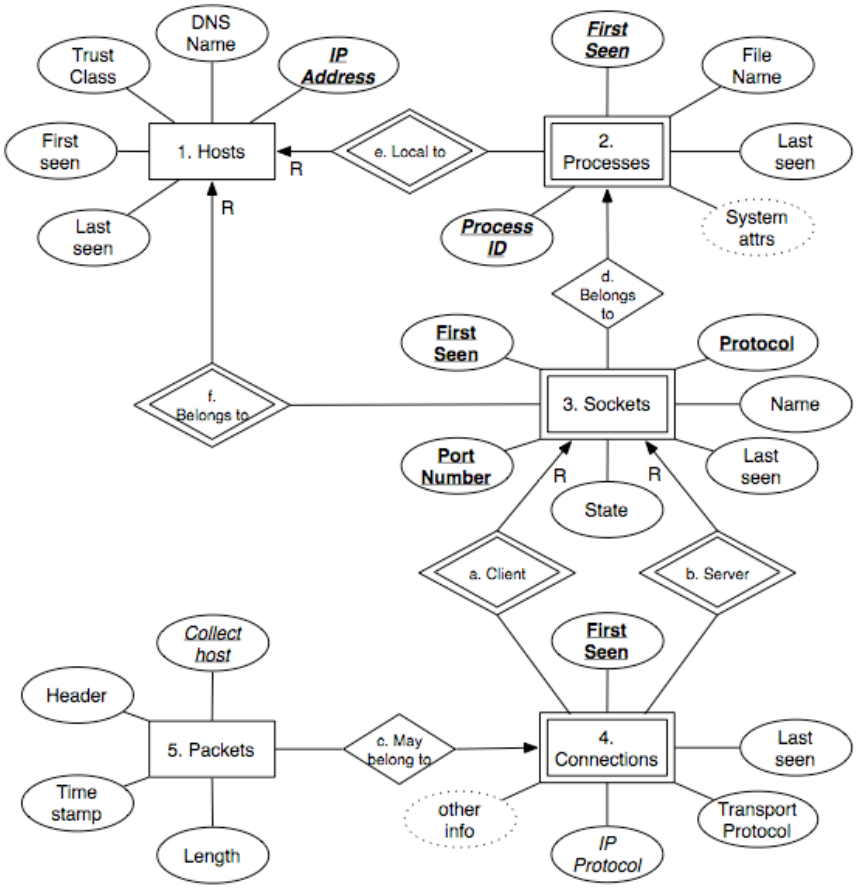


Figure 57: Entity/Relationship diagram of the networked host view domain.

In the following discussion, database entities are often named the same as their real-world components. To distinguish between the two, database entity names will appear in *italics*. We can convert the E/R diagram readily to a set of relations that will form the database schema. Note that HoNe keeps the IP address in two forms: as an integer (intAddr) and in its dotted quad notation (ipAddr). This allows more efficient key lookup in some cases. In each database schema, the names of candidate key elements appear in boldface.

Part of the identification information for every entity in the database is a timestamp of when that entity first appeared. If an entity has only this timestamp, it is called a *point entity* with a zero-length lifespan. For some entities, a second timestamp that is not part of the identifier tells when the entity was most recently seen or updated. These entities are called *duration entities* because they have a greater than zero lifespan. The timestamps are used to reconstruct event histories.

Suppose we wanted to retrieve all processes in existence from time *a* to time *b*. We may write an SQL query to retrieve these as follows:

```
SELECT * FROM processes
WHERE a BETWEEN firstSeen AND lastSeen                    (1)
   OR b BETWEEN firstSeen AND lastSeen                    (2)
   OR firstSeen BETWEEN a AND b                           (3)
```

Figure 58 depicts how this query operates graphically. In Figure 58, the duration entities are shown as rectangles with lifetimes described by their length. The point entities are depicted as diamonds. From the SQL statement above, condition (1) includes database entities 1 and 3, condition (2) includes entities 1 and 5, and condition (3) includes entities 4 and 8.

5.4.4.2 Entities

Hosts: *Host* entities represent machines visible on the network including monitored machines. The primary key for *hosts* is the IP address, which, except for broadcast, multicast, and private addresses, is generally considered unique. HoNe does not treat these other address classes differently. For instance, HoNe treats the network broadcast address as if it were the unique identifier of a particular machine. This will not cause serious problems unless traffic from multiple network segments is brought together for analysis. The IP address of the *host* entity is exported and used as a partial identifier by members of the weak entity sets, *connections, processes,* and *events.*

Hosts also have a Domain Name Service (DNS) name (the text identifier that maps to the IP address), beginning and ending timestamps (derived from the timestamps on packets originating from or destined for that host), and a trust class. HoNe uses four trust classes of hosts: (1) Enterprise-Managed, (2) Enterprise-Unmanaged, (3) Foreign-Managed, and (4) Foreign-Unmanaged. "Enterprise" hosts belong to the same company as the administrator while "foreign" hosts have a different owner. "Managed" hosts are those the administrator (or his group) exercise administrative control over. In most large companies, there will be a sizeable number of hosts that belong to the enterprise, but are not managed by the administrator. "Managed-foreign" hosts are an interesting case, where the administrator manages a machine not owned by the enterprise. Most administrators have personal machines at home, and some are able to use these machines remotely to manage hosts belonging to the enterprise. The *hosts* entity is formalized as Figure 59.

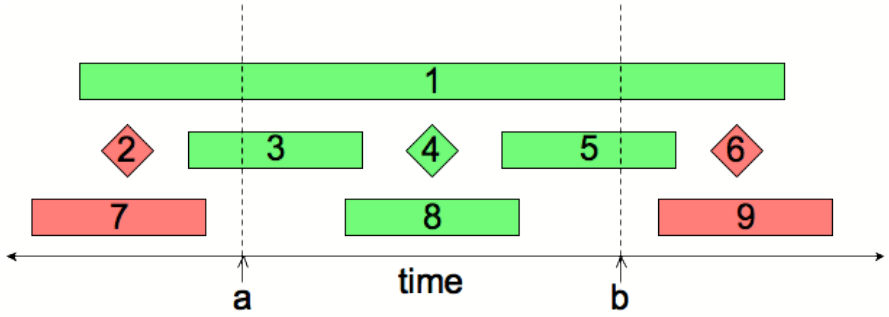


Figure 58: Database entities 2, 4, and 6 are point entities with only a starting timestamp. The others are duration entities with both start and end timestamps. Entities 1, 3, 4, 5, and 8 are considered within time window (a,b).

ipAddr	*intAddr*	*dnsName*	*enterprise*	*managed*	*firstSeen*	*lastSeen*
text	integer	text	boolean	boolean	timestamp	timestamp

Figure 59: The *hosts* relation.

Processes: *Processes* represent programs executing on a monitored host. The primary key for *process* entities is the union of the *host* identifier, the system process identifier (PID), and the timestamp when the first packet owned by this process was seen. Thus, multiple invocations of a process on a host will record multiple *process* records in the database. The PID is not guaranteed to be unique because PID numbers wrap around and start at zero each time the host boots up. Because processes derive part of their identity from the host they run on, they are considered a "weak entity" and symbolized with a double-outlined rectangle. Additionally, *processes* have a file name of the executable, a last-seen timestamp, and system attributes such as owner, cumulative CPU time, etc. The *processes* entity is formalized as Figure 60.

hostRowID	***pid***	***firstSeen***	*name*	*fileName*	*owner*	*perm–issions*	*cpuTime*	*lastSeen*
Host table row id	unsigned integer	timestamp	string	text	text	text	integer	time–stamp

Figure 60: The *processes* relation.

Sockets: A socket, in the technical sense, is simply the association of an IP address and a port number[73]. *Socket* entities represent this association and store some additional useful information about the association. The primary key of the *socket* entity is the union of the *host* to which it belongs, the port number on the host, the protocol used, and the time the socket first sent or received a packet. This arrangement is somewhat different from the technical definition because it is possible for different processes on a host to open up the same socket association at different times. In our database interpretation, two technically identical socket associations will map to separate *socket* entities if the starting times differ. *Socket* entities also store the name of the file or device used to store the socket, the TCP state of the socket (if applicable), and the time the latest event involving the socket occurred. The *sockets* entity is formalized in Figure 61.

hostRowID	***portNumber***	***protocol***	***firstSeen***	*name*	*state*	*lastSeen*
Host table row id	unsigned integer	integer	timestamp	text	text	timestamp

Figure 61: The *sockets* relation.

Connections: *Connections* represent a socket pair and the stream of packets that flows between the two *sockets*: a client, and a server. The primary key of a *connection* is the union of the client and server *socket* keys. For each connection, HoNe stores the IP protocol (e.g., TCP (6), UDP (17), etc.), and the transport protocol number for TCP or UDP (if any). HoNe stores in each *connection* instance a record of whether the connection was set up and torn down normally or not because connections that do not follow the definition of TCP found in [6] may indicate intrusions. Because TCP is not the only protocol of interest, each *connection* has a field that tells whether there has been bidirectional traffic or only unidirectional traffic (as with UDP and ICMP protocols). HoNe also stores a timestamp that stores the most recent time the connection saw activity. This entity is formalized as Figure 62.

clientRowID	***serverRowID***	***firstSeen***	*name*	*ipProto*	*transportProto*	*bidirectional*
Socket table row id	*Socket* table row id	*timestamp*	*text*	unsigned integer	unsigned integer	boolean

…

normalSetup	*normalTearDown*	*lastSeen*
boolean	boolean	timestamp

Figure 62: The *connections* relation.

Packets: Each *packet* is a record of information that crossed the network and was collected by some host. The primary key of each packet is the union of the packet's kernel timestamp and the IP address of the host where the packet was collected. This host address need not map to any *host* entity. HoNe tracks the identifier of the owning connection as well although this information may not always be known. HoNe stores at least the first 68 bytes of packet content (sufficient to capture the IP and TCP headers) in each *packet* object. HoNe stores the total length of the packet to allow bandwidth calculations on the *connection*. This entity is formalized as Figure 63.

timestamp	***collectHost***	*connectionID*	*length*	*header*
timestamp	*text*	*Connections* table row id	unsigned integer	blob

Figure 63: The *packets* relation.

Summaries: One type of entity is not shown in the database entity-relation model and this is the *summary* entity. *Summaries* exist as quick reference objects for the visualization. Summaries are joins of information contained in other entities used to quickly access information and speed visualization. In this way, summaries are like a cache to allow the visualization to respond to queries efficiently. Summaries are derived from the database whenever new connection information is added. The *summary* relation is formalized in Figure 64.

id	***connID***	***firstSeen***	*lastSeen*	*clHostIP*	*clHost–Name*	*clHostMgd*	*clHostEnt*
integer	*Connection* table row id	timestamp	timestamp	*text*	text	boolean	boolean

…

clPort	*clPid*	*clProcName*	*svHostIP*	*svHost-Name*	*svHostMgd*	*svHostEnt*	*svPort*
unsigned integer	unsigned integer	text	text	text	boolean	boolean	unsigned integer

…

svPid	*svProcName*	*style*
unsigned integer	text	text

Figure 64: The *summaries* relation.

5.4.4.3 Relationships

Every *connection* must have exactly one client *socket*: Since we are mostly concerned with TCP, a client process initiates each connection via a socket. *Connections* cannot be added without a client *socket*. This relationship is not formalized because the primary key of the *connection* is partly formed from the *socket's* primary key.

Every *connection* must have exactly one server *socket*: The same rationale applies to server *sockets*. This relationship is also not formalized.

Every *packet* may belong to up to one *connection*: Packets may be observed and stored in the database without belonging to any *connection*, but they may belong to no more than one *connection*. The stream of packets becomes part of the connection history. If a packet belongs to no known connection, then HoNe generates a new connection from the source and destination IP addresses and port numbers within the packet. That connection will have an unknown client, server, or both. This relationship is not formalized.

Every *socket* belongs to up to one *process*: Processes own sockets. In our model, a socket may not belong to more than one process, although in practice it is possible for separate processes to share the same socket. We assume the socket creator is its sole owner. This relationship is formalized as Figure 65.

processRowID	***socketRowID***
Process table row ID	*Socket* table row ID

Figure 65: The *socket belongs to process* relation

Every *process* is local to exactly one *host*: All processes run on some host. This relationship is not formalized because the host identity is part of the process's identity.

Every *socket* belongs to exactly one *host*: All belong to some host. This relationship is not formalized because the host identity is part of the socket's identity.

I designed the database for quick retrieval of blocks of data by the visualizer. When the user loads a data file, sets a time range, or executes a query, numerous database activities are triggered behind the scenes. Thus, the visualizer may be thought of as a graphical front-end to the database. The responsiveness of the visualization depends to a large degree upon the responsiveness of the underlying database.

5.5 Summary

The HoNe Visualizer and its supporting architecture provide a first glimpse across the host/network divide. No other tool has been able to visualize a correlated view of packets and the processes responsible for them. In this section I have shown how the host/network divide is located in the kernel and that the bridge must be located there too. In this chapter, I have introduced my visualization of the correlated data and shown how it works in an actual intrusion of a monitored host. I have also explained the architecture of HoNe and why I chose the implementation I decided upon. In the next chapter, I will present my usability evaluation of HoNe that demonstrates how HoNe helps administrators perform better on intrusion detection tasks.

CHAPTER 6 EVALUATION

My objective for developing HoNe was to show that integrating host and network data into a single visual display provided useful insight for experienced system administrators trying to investigate anomalous behavior. The host-network integration was in two parts: (1) correlating data from the separate sources into one new source, (2) aggregating the new data from voluminous text to a compact visual form. After building HoNe, I evaluated it to find out how well my objectives were met and what remained to be done.

6.1 Research Questions

I wanted to verify that HoNe met the needs that I had identified. I also wanted to determine whether use of HoNe resulted in better intrusion detection performance by system administrators. If use of HoNe resulted in better performance, I wanted to know why. I designed the evaluation to answer the following questions:

1. Does visual packet-process correlation enhance intrusion-detection insight over tools currently in use?
2. What are the benefits and pitfalls of visual and textual presentations?
3. What are the benefits and pitfalls of packet-process correlated and noncorrelated presentations?

I was also curious to a lesser degree to find out what level of experience users needed to benefit most from HoNe. This section discusses the two-phase usability evaluation: pilot interviews and summative usability evaluation. The purpose of the pilot interviews was twofold: (1) to bring to light missing parts of the visualization, and (2) to determine how much intervention would be needed during the summative usability evaluation. The purpose of the summative usability evaluation was to quantitatively answer the research questions.

6.2 Pilot Interviews

6.2.1 Participants

I conducted a pilot study with six expert computer security professionals. I selected them from the set of previous experiment subjects based on their known expertise and because of their helpfulness in prior interviews. I asked the pilot study subjects to honestly evaluate the visualization as I ran through a scenario of discovering a hacking incident. They provided valuable insight helping me determine what pieces were still missing from the visualization and how much intervention would be necessary for the less experienced users in the study to be able to use HoNe productively.

I showed the visualization to each pilot subject using real data collected from the kernel modifications. Additionally, I showed the visualization to two information visualization experts who provided a few other helpful comments.

6.2.2 Findings

From the pilot study, I learned that HoNe was a great improvement over Portall. The pilot subjects had comments that fell into four categories: (1) enhancement requests (26 items listed in Table 9), (2) areas where intervention would be needed (6 items listed in Table 10), (3) negative comments (8 items listed in Table 11), and (4) positive comments (11 items listed in Table 12).

Table 9: Enhancements requested by pilot interview subjects

Issue	Resolution
Display all times in human-readable formats (not seconds since the epoch).	Fixed.
Add a scrollbar to the filter list.	Deferred. Workaround: List autoscrolls.
When a host icon is clicked show whois results.	Fixed.
When a port is clicked, search trojaned ports list for matches (2 subjects).	Partially fixed. The port information in /etc/services file is now displayed.
List open files for processes.	Rejected. Requires too much data collection and storage.
Enable horizontal magnification of the time windows so you can see what is going on (3 subjects).	Deferred for future work.
Add a legend so people will know what they are looking at.	Rejected. Training issue to be handled via intervention. No screen space for a legend.
Add an animation of a clock to show when the application is busy.	Partially fixed. The status line now indicates when the application is busy.
Minimize more info and filters panes to get bigger overview space.	Fixed. More info and filters panes combined into a single tabbed pane.
Expand use of tooltips (2 subjects).	Fixed. Added more tooltips, etc.
Line width should indicate bandwidth.	Deferred for future work. Line widths are customizable by the user.
Make it real-time.	Deferred for future work.
Full packet grabbing of certain types of packets.	Deferred for future work.
Add support for ICMP and UDP.	Deferred for future work.
Let the user designate trust class of hosts by dragging and dropping them into another region.	Deferred for future work.
Log file correlation/visualization would be helpful, but not necessary at this point.	Rejected. Out of scope.
Need some kind of alert mechanism that tells you when you need to look for something.	Rejected. Out of scope.
Want to be able to export data into tcpdump format for co-processing with other tools like TcpView.	Deferred for future work.
Add ability to export pictures of the data.	Deferred. Workaround exists.
Calculate windowed inter-arrival time for sessions and services by port.	Rejected. Out of scope.
Summary data for older periods, roll-up into hierarchy of bins.	Deferred for future work.
Add some way to control database size.	Deferred for future work.
Need some way to search for specific problems in the large overviews.	Fixed. May now highlight all connections that match a given filter.
Add interactions like highlighting all connections terminating in a given selected port.	Fixed.
Use 3D so you can fit more flows in.	Rejected. Out of scope.
Reuse established GPL code to do process name lookups, etc.	Rejected. Kernel mod works as is.

I used feedback from the pilot subjects to refine the user interactions and added options to provide more information about hosts, processes, and ports as the experts requested. I found that the area needing the most intervention was constructing SQL queries. I also found that my initial implementation of time windows based on slider widgets was unusable, so I replaced it with a more graphical approach that employs histograms and connection lines.

Table 10: Interventions needed as identified by pilot interview subjects

Intervention	Status
Writing SQL expressions for filters is a special skill beyond the scope of much of my user population (expressed by 4 subjects).	Deferred for future work.
Explain why *all* the processes don't show up (not just the communicating ones).	Fixed via interview protocol.
Explain why some of the processes aren't named.	Fixed via interview protocol.
Managed, Enterprise, Foreign paradigm needs to be explained.	Fixed via interview protocol.
Sometimes it is difficult to know which forked subprocess a particular process belongs to.	Deferred. Kernel mod is unaware of process hierarchy.
Tool requires more expertise than the average system administrator has.	Fixed via interview protocol.

Table 11: Negative features cited by pilot study subjects

Feature	Status
Don't use red except for danger.	Fixed. Eliminated red unless selected as a filter color by the user.
Rethink entire time window presentation.	Fixed. Time window replaced with a direct manipulation widget.
Filter number column is confusing since filters can't be reordered.	Deferred. Removed filter number column and ability to sort on column. Filters are processed in the order they are listed.
Users won't like the small text size in the more info window.	Deferred. Easy fix, but impact on screen space is too great.
Confusion over role of histogram bars and connection lines.	Fixed via interview protocol.
Packet-process correlation will not help in cases where the affected process is just a vector to cause another process to misbehave.	Rejected. Temporal correlation of these process activities will still yield some insight.
Should be using a more recent version of the kernel.	Rejected. The kernel mod works fine for testing purposes.
Visualization should be redesigned to handle much higher bandwidth (on the order of 800 flows per second).	Rejected. Out of scope.

Of the 38 issues raised that indicated some change that needed to be made to the visualization or the correlation engine, I fixed 15, deferred 13, and rejected 10. The deferred items are generally fixes I agree with but that were too difficult or unnecessary to proceed with testing. The rejected items were generally ideas the interviewees had that were contrary to the intended purpose of the tool.

After I made the fixes suggested by the pilot interviewees, I was ready to begin planning the summative usability evaluation. First I designed the experiment and then collected data to be used in the experimental scenarios. The next sections describe these phases of the work.

Table 12: Positive features cited by pilot study subjects

# Subjects	Feature
1	Filters are acceptable with usability intervention.
2	SQL backend and filters considered powerful.
1	New time window is much improved.
1	Nothing else does packet-process correlation.
1	Graphical depiction of individual connections is good.
2	Would be good for detecting zero-day exploits.
1	Information in the packet dump is all you really need to know.
1	Whois lookup of hosts considered useful.
1	Correlation is a new and unique data source.
1	Helps identify when further investigation is necessary.
1	The visual correlation is a small step in the right direction and provides a little benefit over tcpdump, ethereal, etc.

6.3 Summative Usability Experiment

6.3.1 Experiment Design

When I designed the experiment, I intended to determine whether visualization, packet-process correlation, or some combination of the two had the greatest effect on insight. Thus the specific question I was seeking to answer was, "Is user insight a function of correlation, visualization or both?" The hypothesis I was testing was as follows:

Hypothesis test: H_o: The factors do not influence the insight gained
H_a: One or more of the factors do influence the insight gained

I designed a two-factor experiment, with the primary factors being (A) visualization—at two levels (on or off), and (B) correlation—at two levels (on or off). A secondary factor could be scenario, but I was not interested in determining the differences in the amount of insight gained between scenarios. Thus I used a 2x2 (visualization vs. text data only and correlated vs. uncorrelated), within-subjects design. I planned for 16 subjects. The four experimental conditions were thus:

1. No-Visualization/No-Correlation (NVNC): User has no visualization and only the uncorrelated output of tcpdump, netstat, and lsof to work from (Control condition).
2. No-Visualization/Correlation (NVC): User has correlated data from the kernel mod plus the control data.

3. Visualization/No-Correlation (VNC): User has separate visualization windows for netstat and tcpdump data. All the textual control data is available.
4. Visualization/Correlation (VC): User has all the data and a visualization of the correlated data.

I chose within-subjects design because the population of qualified subjects is rather small and getting subjects was difficult. There were four experimental conditions and four scenarios (*i.e.*, datasets). I had each subject perform under each condition varying the order to counter any learning effect. I also varied the order that I assigned scenarios to conditions so that each subject saw each scenario exactly once and so each (condition, scenario) pairing appeared in each testing order exactly once. In each scenario, I asked the subjects to identify evidences of intrusions and any other interesting feature of the data. I established "ground truth" in conjunction with recognized experts in computer security and scored the participants on how much of the truth they discovered in each case.

6.3.2 Scenarios

I collected data for five scenarios (four and a backup in case any data was corrupted), one of which I assigned in block fashion (without repetition) to each condition. For each scenario, I simultaneously collected data from netstat, lsof, tcpdump, syslog, and my kernel modifications. The netstat polling period was set to 1 sec, the lsof polling period was 60 sec, and all other data collection was continuous. Each scenario was at least 12 hours long. The following is a description of each scenario

1. Control: Linux RedHat Fedora Core 3 system running only an SSH server, no engineered attacks. (Normal configuration)
2. Engineered Hack: Under cover of an SSH scan, an intruder logs in to an unprivileged account, downloads a rootkit from a remote server, and starts a new Internet service.
3. Normal: As control, but with attacks that do not result in intrusions.
4. Uncontrolled Hack: Similar to the engineered hack above but real, not engineered.

I collected data for an additional sample scenario to be shown for training and to be used as a backup in case any of the data in any condition was corrupted. Table 13 shows the assignment of conditions and scenarios to subjects. Because 16 subjects would cover all combinations of condition and scenario, I decided not to assign condition and scenario randomly. By varying the scenarios across conditions I hoped to avoid problems with any scenario being inherently easier or harder. In the end, I was able to get 27, self-selected subjects with a wide range of experience from between 0 to 20 years (mean 3.79, stdev 6.55). I simply perturbed the order of conditions and began assigning scenarios for subjects 17-27 as for subjects 1-11. Statisticians from the statistical consulting group and an independently consulted statistician both verified that the experiment was balanced despite the lack of random assignment. While random assignment would have been superior, my assignment method did not harm the analysis of the results.

Table 13: Assignment of conditions and scenarios to subjects.

Conditions: 1=NVNC, 2=NVC, 3=VNC, 4=VC

Subj #	Condition, Scenario Pairing			
	Run 1	**Run 2**	**Run 3**	**Run 4**
1	1, 1	2, 2	3, 3	4, 4
2	2, 1	3, 5	4, 3	1, 4
3	3, 1	4, 2	1, 3	2, 4
4	3, 1	4, 2	1, 3	2, 4
5	4, 1	1, 2	2, 3	3, 4
6	1, 2	2, 3	3, 4	4, 1
7	2, 2	3, 3	4, 4	1, 1
8	3, 5	4, 3	1, 4	2, 1
9	4, 2	1, 3	2, 4	3, 1
10	1, 3	2, 4	3, 1	4, 2
11	2, 3	3, 4	4, 1	1, 2
12	3, 3	4, 4	1, 1	2, 2
13	4, 3	1, 4	2, 1	3, 5
14	1, 4	2, 1	3, 5	4, 3

Subj #	Condition, Scenario Pairing			
	Run 1	**Run 2**	**Run 3**	**Run 4**
15	2, 4	3, 1	4, 2	1, 3
16	3, 4	4, 1	1, 2	2, 3
17	4, 4	1, 1	2, 2	3, 3
18	4, 1	3, 5	2, 3	1, 4
19	1, 1	4, 2	3, 3	2, 4
20	2, 1	1, 2	4, 3	3, 4
21	3, 1	2, 2	1, 3	4, 4
22	4, 2	3, 3	2, 4	1, 1
23	1, 2	4, 3	3, 4	2, 1
24	2, 2	1, 3	4, 4	3, 1
25	3, 5	2, 3	1, 4	4, 1
26	4, 3	3, 4	1, 2	2, 1
27	1, 3	4, 4	3, 1	2, 2

I wrote a program to ensure fairness in assigning conditions, scenarios, and orderings to subjects. The program compensates for a data anomaly I discovered during the evaluation. Figure 66 contains a listing of the Tcl code I used to assign conditions and scenarios to subjects.

6.3.3 Scoring

For each scenario I derived a list of features and assigned each feature a unique identifier. Subjects gained points for noticing and correctly diagnosing features but lost points for incorrectly diagnosing a noticed feature. The possible diagnoses of each feature were:

MP = Malicious penetration: Evidence that the machine has been compromised.
NM = Non-Malicious: Activity is normal behavior on the machine.
NP = Non-Penetrating: Malicious activity that shows no evidence of having compromised the machine.
- = Subject did not notice this feature (always zero points)

For each scenario and viewing condition, I added the positive and negative scores separately. Larger absolute score values indicate more features were noticed. Higher positive scores indicate that the subject more often correctly diagnosed features he/she noticed while higher (absolute) negative values indicate he/she tended to misdiagnose features. Scores closer to zero indicate fewer features were noticed. Research questions this scoring was designed to answer are:

1. Do the viewing conditions cause people to notice more features?
2. Do the viewing conditions help people to correctly diagnose the features they did notice?

```
#!/bin/bash
# Next line starts tclsh \
exec tclsh "$0" "$@"

set case 1
for {set i 0} {$i <= 3} {incr i} {
    for {set j 0} {$j <= 3} {incr j} {
        puts -nonewline [format "%d\t" $case]
        for {set s $i; set c $j} {$s <= $i+3 && $c <= $j+3} {incr s; incr c} {
            puts -nonewline [format "%d, %d\t" [expr round(fmod($c,4) + 1)] \
                [expr round(fmod($s,4) + 1)]]
        }
        incr case
        puts ""
    }
}

for {set i 0} {$i <= 3} {incr i} {
    for {set j 0} {$j <= 3} {incr j} {
        puts -nonewline [format "%d\t" $case]
        for {set s $i; set c [expr $j+3]} {$s <= $i+3 && $c >= $j} {incr s; incr c -1} {
            set cond [expr round(fmod($c,4) + 1)]
            set scen [expr round(fmod($s,4) + 1)]

            # Compensate for a known bad data combination
            if {$cond == 3 && $scen == 2} {
                set scen 5
            }
            puts -nonewline [format "%d, %d\t" $cond $scen]
        }
        incr case
        puts ""
    }
}
```

Figure 66: Tcl code for assigning (condition, scenario) pairings.

I normalized the positive and negative scores by dividing the subjects' positive scores by the maximum possible score for the scenario used and dividing the negative scores by the minimum possible score (Figure 67). I used the normalized scores to allow me to compare scores across different scenarios. Then I subtracted the normalized negative score from the normalized positive score to obtain the *insight score* (Figure 68). This approach prevents a user from attempting to increase his score by reporting lots of inconsequential features and reduces the likelihood of guessing. I used the insight score to award prizes among other things.

6.3.4 Response Variable

The response variable is amount of insight gained. I designed the *insight score* to quantify insight gained. As mentioned earlier, I catalogued the features of each dataset before scoring the subjects. I established "ground truth" about what happened in the scenarios using my *a priori* knowledge of the situation and the judgment of security experts using all the available data (including the syslog files not made available to the experimental subjects). Features may be either benign or malicious (indicating an intent to gain improper access). Malicious features may be further classified as penetrating (giving the attacker use of the machine) or nonpenetrating (e.g., a script kiddie attack that just bounces off the firewall). Depending on the viewing condition, some features appeared (given all available evidence) to be penetrations when they were not or *vice-versa*. These I term *apparent penetrations* or *apparent nonpenetrations* and treat as actual penetrations or nonpenetrations respectively in the viewing

conditions where they appear. An example of an apparent penetration is when a bug in the bridge process lumped several short SSH connections together so that they appeared as one long connection. Given the user's viewpoint and all the data available in that viewing condition, it would be reasonable to assume that the long connection implied that a person successfully logged in even though, in actuality, there were only a series of short attempts. I assigned points to each possible diagnosis for each feature using the seven-point scale shown in Table 14.

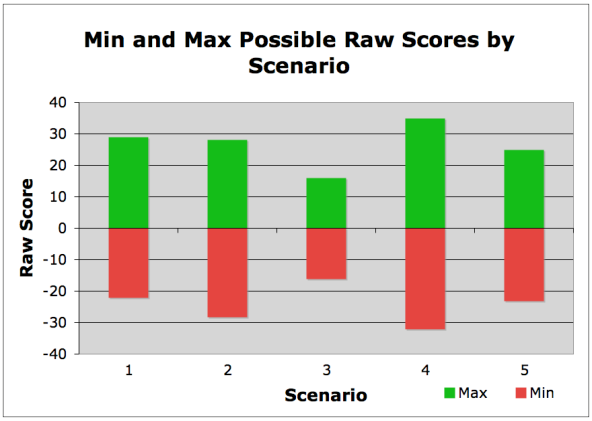


Figure 67: Minimum and maximum raw scores attainable in each scenario.

$$Insight = \frac{Noticed}{\|Noticed\|} - \left|\frac{Misdiagnosed}{\|Misdiagnosed\|}\right|$$

Figure 68: Formula for Insight Score.

I assigned a one point of extra credit when a subject identified a malicious penetration (real or apparent) I had not previously catalogued or noticed that a feature was an artifact (e.g., that it was engineered, inconsistent, etc.). The actual number of features that occurred during the monitored periods is not truly knowable and is somewhat subject to the judgment of the observer, hence the extra credit category.

Table 14: Insight Score scoring guidance.

Score	Meaning
-3	Diagnosing a benign feature as a malicious penetration or Missing a major malicious penetration
-2	Diagnosing a malicious nonpenetration as a malicious penetration or Missing an apparent penetration (given the condition) or Missing additional major malicious-nonpenetrating features beyond the first
-1	Diagnosing a benign feature as a malicious nonpenetration or Missing a major malicious nonpenetration
0	Noticing a benign feature
+1	Properly diagnosing a malicious nonpenetration or Properly diagnosing a malicious penetration without supporting reasoning (guessing)
+2	Properly diagnosing a malicious penetration
+3	Properly diagnosing and assessing the impact of a malicious feature (real or apparent) or Properly noting the relationship of two or more malicious features together

Although performance-time data would seem to be a good measure of insight, I decided not to collect it because good performance is a function of accuracy more than time in this case. The kinds of tasks I was asking the subjects to do were open-ended, ill-conditioned, and ill-defined and are thus not easily measured with simple measures such as elapsed time. Based on my own system administration experience, interviews with professional system administrators, and my understanding of the literature[74], timing would provide little useful information. Accuracy is the most important characteristic of the kind of work I am studying. If administrators quickly reach an incorrect conclusion the consequences could be more costly than if they reach a correct conclusion more slowly than desired. Thus, I consider timing data as potentially interesting, but not critical to the study.

Each iteration of the experiment required about 90 minutes of time with the participant. Many of the experiments I was able to conduct in a laboratory setting, but a few of my subjects, being professionals on the job, were unable to participate unless I came to them. I showed each participant four datasets where the monitored computer may or may not have been hacked. The participant could use tools appropriate to the viewing condition (tcpdump, ethereal, output from netstat and lsof, and HoNe) to diagnose what actually happened to the machine. Subjects had approximately 15 minutes to diagnose features in each dataset.

Originally, I had planned to test only experienced system administrators, but then I had the opportunity to test 23 students from a computer security class, about half of whom had some administrative experience. Because the subjects varied greatly in level of expertise and experience, I provided help in basic knowledge and skill areas such as how to write SQL database queries or tcpdump packet filters. I also provided brief training on all needed skills and told each subject the limitations of how much I would help them. The reason for this intervention and training was to bring all the subjects up to a certain minimum skill level to allow them to complete the scenarios. I was not interested in testing how well people used basic tools such as tcpdump and grep. I was interested in how well various viewing conditions helped them see what was happening in a dataset. I provided the same training and intervention for all subjects.

6.3.5 Data Collection

To generate data for the visualization, I set up a sacrificial host on the campus LAN and outfitted it with Snort and Tripwire (two freely available intrusion detection systems). Then I created a User Mode Linux (UML)[75] virtual machine and equipped it with the modified kernel running in a process on the real machine. I bridged the real machine's Ethernet interface with that of the virtual machine, so that from the outside they looked like separate machines on the same LAN segment. During the evaluations, I wanted interviewees to be responding to real data from actual break-ins, so I purposely weakened the virtual machine's defenses by running old versions of Apache, MySQL, and RPC without any firewall. I hardened the real machine by only allowing SSH connections to it, but I disabled the firewall since that interfered with the virtual host's communications. Eventually, after waiting for weeks for hackers to break in, I had to resort to extreme measures such as a "guest" username with the password "guest" to ensure the machine would be hacked in a timely manner. Such measures almost always resulted in a successful break-in within 12 hours. For instance, one hacker broke in via this unprivileged account and set up an IRC peer disguised as a Network Time Protocol daemon.

The virtual machine named "Moe" was the monitored machine for all scenarios. This machine was running Red Hat Fedora Core Linux (FC3 v2.6. 13-1). The host machine, "Green" was running Debian Linux (Sarge, v2.6.8). I set up the Ethernet interfaces on the host so that the two machines looked like peers on the same network segment from the outside (see Figure 69). Using UML allowed me to let

the virtual machine become hacked without compromising any real machine and made restoring the machine to original conditions as simple as deleting the memory image stored in a file on the host machine.

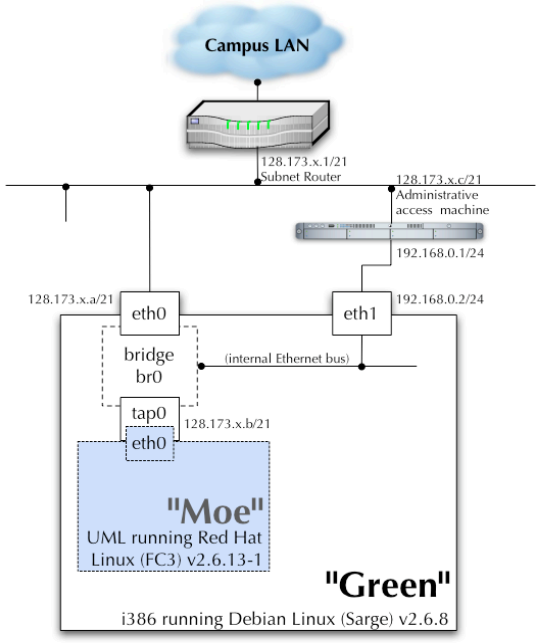


Figure 69: Virtual host setup for gathering scenario data.

The process of data collection for each scenario was the same although the kinds of attacks performed were different. For the scenario numbered X, I performed the following steps on the virtual machine:

1. Record all administrative activities in /root/scenarioX/scenarioX.log.
```
mkdir /root/scenarioX
cd /root/scenarioX
cat >> /etc/syslog.conf
*.debug,mark.debug /root/scenarioX/scenarioX.fulllog
/etc/init.d/syslog restart
logger “Starting setup for scenario #X”
```
2. Run netstat, lsof, and tcpdump as follows:
```
nice /bin/netstat --tcp --listening --all --numeric  --extend --extend --
verbose --program --continuous > scenarioX.netstat 2>&1 &
nice /usr/sbin/lsof -n -r 60 > scenarioX.lsof 2>&1 &
nice /usr/sbin/tcpdump -i eth0 -w scenarioX.tcpdump \(tcp or udp \) and
host localhost > /dev/null 2>&1 &
```
3. Remove any existing /root/iptab/ipt_pid log files, then start the kernel mod via: insmod /root/pid.ko.
```
rm -f /root/iptab/ipt_pid
insmod /root/pid.ko
ln /root/iptab/ipt_pid scenarioX.iptab
```
4. Allow the system to run for at least 12 hours. Perform any engineered hacking activities.
5. Log in to the virtual machine, stop netstat, tcpdump, and the kernel mod, and harvest their output files.

```
rmmod pid.ko
killall netstat
killall tcpdump
killall lsof
cd /root
tar cvjf scenarioX.tbz scenarioX
```

6. Transfer the archived output files to a remote machine for processing. Convert the kernel mod, netstat, and tcpdump output files to database files via the bridge processes. Convert the binary kernel mod output file to text and store all other text files as-is for use by the test subjects.
7. After running each test, discard the UML's "Copy on Write" file from the host machine that contains all changes made to the virtual machine for this scenario and start from scratch. This destroys any hacking or configuration changes that were actually done to the machine.

I collected data for more than 10 different scenarios and selected the best of the lot to use in the study. I found that a defect in the kernel modifications caused the machine to crash when the attacker made certain modifications to the machine. Kernel-level rootkits such as Adore[76] and LRK5[77] caused the machine to crash when they were activated. Thus, after the attacker initially penetrated the machine, usually his attempts to install a rootkit would crash the virtual machine. A positive result of this "feature" was that hackers were unable to use the machine to exploit other machines on the campus network. Unfortunately, however, the defect prevented me from collecting data beyond the initial breakin. Additionally, some methods of breakin also crashed the machine, so many of the scenarios were unusable. Once all the scenario data files were collected, I was ready to begin the summative usability evaluation

6.3.6 Analysis

This section discusses the quantitative findings from the experiment. I will first explain the problems I experienced with the design and conduct of the experiment and discuss their impact. Next I will present the quantitative and qualitative findings from the experiment and discuss the implications of these results for the utility of the technology.

6.3.6.1 Problems with the experiment

Problems with the study prevented me from deriving the full benefit of statistically rigorous quantitative results as planned. As mentioned earlier, I did not randomly assign scenarios to conditions based on the assumptions that (1) having 16 cases, one for each combination of condition and scenario was sufficient, and (2) that differences among scenarios were insignificant. The first assumption was violated when I received more than 16 subjects and fewer than 32. Thus, not every combination was equally represented. A more robust approach would have been to assign the conditions and scenarios randomly to each subject. The second assumption turned out to be unwarranted because the insight scores for the various scenarios were significantly different (p-value 0.0002; see also Figure 70). My statistical consultants have verified that the experiment was properly balanced so that the effects of these defects are not fatal.

Another problem I encountered in assigning conditions and scenarios was that some of the netstat data for scenario 2 would not display properly in the visualization. Thus, I was unable to assign scenario 2 when viewed under condition 3 (VNC). For these cases, I substituted scenario 5 data. This substitution also resulted in a defect in the experiment complicating analysis and decreasing certainty of the quantitative results for the VNC condition. As mentioned earlier, a few problems with the visualization of the netstat data under one scenario caused some modification of the blocking plan for the assignment of scenarios to conditions. This defect impacted the statistical reliability of the results.

Another problem with the experiment that did not surface until the results were in was that users' performance varied more widely on the VNC condition than the others (see Figure 71). I suspect that the reason is the visualization used for the VNC condition was identical to that used for the VC condition, but the data was not correlated. People found this confusing and misleading.

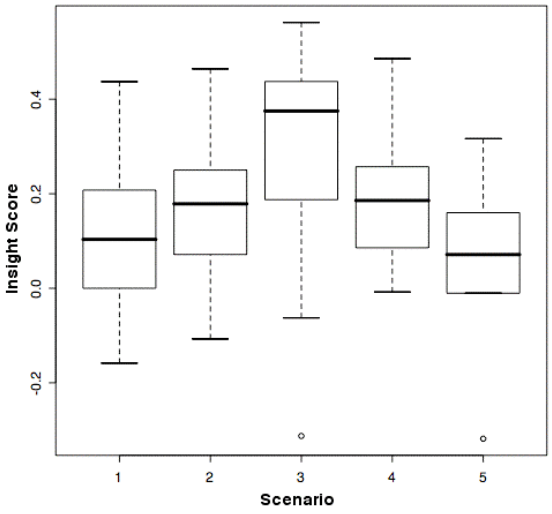


Figure 70: Box and whisker plot of insight scores by scenario showing that scenario 3 had higher scores than the other scenarios.

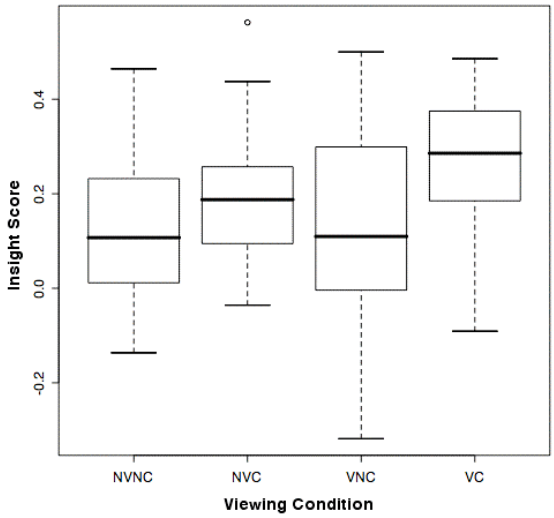


Figure 71: Box and whisker plot of insight scores by viewing condition showing the greater variation for the VNC condition.

Thus, it was not a simple matter to separate the effects of visualization and packet-process correlation. An example of the confusion inherent in the VNC condition is illustrated in Figure 72. The figure shows what happens when uncorrelated tcpdump data is shown in the HoNe visualizer. Every packet that is seen on the network, whether accepted by the monitored host or not, looks like a completed connection, because without correlation to an internal host view, HoNe cannot determine which packets affect the internal state of the machine and which are refused. Users found these false connections confusing and this resulted in the VNC condition producing much less insight in some cases.

The corresponding netstat view would show none of these false connections, often further confusing users. Users of the VNC condition would sometimes conclude that the two views (netstat and tcpdump) were contradictory rather than complementary to one another. Thus many users chose one or the other as their primary view and made most of their judgments based on that one view. The experiment design rewarded identifying features that affected the internal state of the machine, so users that selected the netstat view often performed very well while those that relied exclusively on the tcpdump view did very poorly. I believe this behavior I witnessed while observing the experiments accounts for a large amount of the variation in the VNC scores.

As mentioned previously, a data corruption error forced me to substitute scenario five data in all the cases where VNC would have been used to view scenario two. Scenario five scores were actually somewhat lower than scenario two scores, further eroding my confidence in the cases employing the VNC condition.

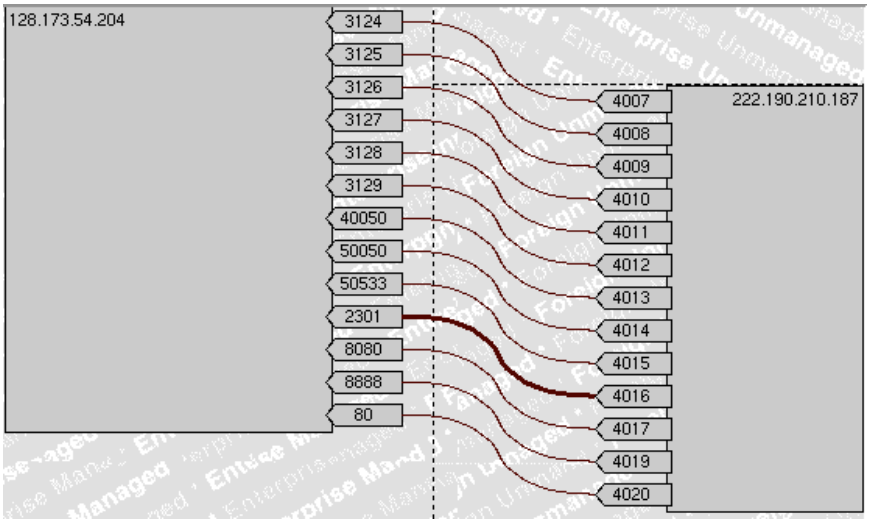


Figure 72: HoNe Visualizer showing "connections" from a foreign unmanaged machine. Actually these lines are showing connections refused by the machine. The appearance of these false connections was confusing to study participants.

6.3.6.2 Quantitative Findings

The combination of these two problems led me to change the way I analyzed the results from a 2x2 matrix to a single factor with three levels: NVNC (control), NVC, and VC. This eliminated systemic problems with the troublesome VNC condition while still allowing me to determine whether either of the remaining treatments were helpful. However, resorting to this analysis cost me the ability to measure the effects of visualization and packet-process correlation independently. Because of this major change from the experiment design and because of the problems cited in section 6.3.6.1, I enlisted the help of Virginia Tech's Statistical Consulting Center. The following discussion is their contribution to my work.

There are twelve treatment combinations in the modified experiment (four scenarios by three conditions). We postulated that different subjects in my study would probably respond to each treatment combination differently. Under this assumption, it was appropriate to use an ANOVA model that estimated the variability in both the treatment combinations and in the subjects. This model captured all subject-to-subject variability so that it would not be classified as unexplained error variability.

Upon fitting this model the statisticians found that my rotating assignment method for the scenarios and conditions was somewhat inefficient, because it did not preserve the ability to quantify interactions between the scenarios and the viewing conditions. The inefficiency was partially a result of having to remove the troublesome VNC condition and partially because I assigned (condition, scenario) pairings to subjects instead of assigning viewing condition and scenario completely independently via some random process. However, even with this inefficiency, the model demonstrated that the control condition (NVNC) was statistically inferior to the others.

Ideally, by estimating the variability in both the treatment combinations and subjects, the model would let us see whether the same viewing condition was best no matter which scenario was used. However, the inefficiency in my assignment of treatment combinations again prevented us from even determining the amount of variability due to scenario, and this prevents us from performing a statistical test to see whether the scenario actually contributes to the variability in the insight score. Thus, we

settled on a slightly different model that assumes that the different conditions would have roughly the same effect on the insight score regardless of scenario. This new model adds to the previous one by taking subject identity into account along with viewing condition and scenario numbers.

We compared the two models using an approximate statistical procedure that tested our assumption that conditions have same effect no matter which scenario is given. The null hypothesis stated that there was no interaction between scenario and viewing condition. To restate it more plainly, the null hypothesis was that the score under any given scenario did not depend on the viewing condition, and under any given condition the score did not depend on the scenario. The statisticians could not reject the null hypothesis via the model comparison test. They estimated the p-value of the test to be near 0.5. Thus we conclude that our assumption that different viewing conditions would have roughly the same effect on the insight score regardless of scenario is reasonable.

Table 15 shows the test results from the new model showing that there is a statistically significant difference between scores from different treatment combinations when taking subjects, conditions, and scenarios into account. Estimating the subject-to-subject variability costs 26 degrees of freedom (df). Since we assumed that the subject-to-subject variability is not important, we did not test whether the difference among subjects was statistically significant.

Table 15: Overall model results taking subjects, conditions, and scenarios into account.

	df	**SS**	**MS**	**F**	**p-value**
Model	31	1.256799	0.040542	2.62	**0.0013**
Error	49	0.759168	0.015493		
Total	80	2.015967		**R-squared** = 0.623422	

Since there are four scenarios, we use 3df to estimate the contribution of scenario to the model. The test for whether scenario affects insight score gives SS=0.376979 and MS=0.125660, with F=8.11 and a p-value of 0.0002, meaning that subjects did better on at least one of the scenarios.

Given the three viewing conditions (NVNC, NVC, and VC), we use 2df to estimate the contribution of viewing condition to the model. The test for whether viewing condition by itself affects insight score gives SS=0.261358 and MS=0.130679, with F=8.43 and a p-value of 0.0007, meaning that subjects did better on at least one condition.

One nagging question that we wanted to answer was whether the experimental unit was the individual subject or the subject-treatment combination. If the former, then we would only have 26df, but if the latter, we would have 80df giving our tests much greater statistical power. We determined that the experimental unit was the subject-treatment combination because each of the 27 subjects experienced three different conditions (after discarding the problematic VNC condition). I did not obtain one overall score for each of the 27 subjects (which would result in 26 total df), but rather I got one score each time a subject faced a new condition. Therefore, the experimental unit was not the subject but rather the subject-treatment combination. This is one of the reasons we used a model that removed the subject-to-subject variability from the rest of the error. Our procedure was confirmed with a faculty member in the Statistics department.

Armed with a model and justification for the number of degrees of freedom, we were ready to analyze the data. Figure 73 illustrates the mean insight scores by scenario while Table 16 shows the statistical probability that there is no significant difference between the mean insight scores of any two

given scenarios. As before green, bold-face p-values indicate significance at the 0.05 level or better while yellow, italic p-values indicate significance at the 0.1 level.

Table 16 shows that scenario three has a statistically significant higher average score than either scenario one or scenario two, and that scenario three has a marginally statistically significant higher average score than scenario four. I conclude that scenario 3 was "easier" than the others probably because there were fewer raw points available in this scenario (see Figure 67). There were fewer raw points because it was the shortest scenario with the fewest events happening in it, and none of the events were complex nor were there any compromises. In fact, I noticed that in scenario 3 most subjects could rapidly rule out the presence of malicious penetrations and thus spent the least time on it. This explains the higher score on this scenario.

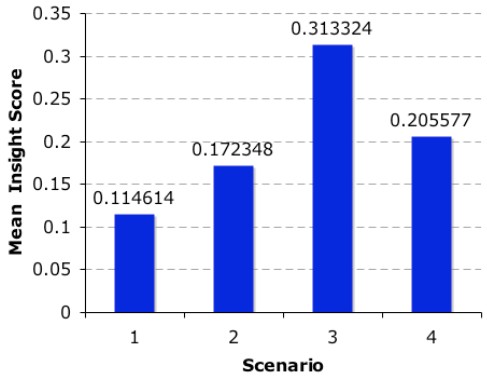


Figure 73: Comparison of mean insight score by scenario.

Table 16: Pairwise probabilities of no significant difference between the mean insight scores by scenario.

Between Scenarios		Difference of Mean Insight scores	P-value
1	2	-0.057734	0.5062
1	3	-0.198711	**0.0001**
1	4	-0.090964	0.1720
2	3	-0.140976	**0.0046**
2	4	-0.033229	0.8526
3	4	0.107747	*0.0635*

Figure 74 illustrates the mean insight scores by viewing condition while Table 17 shows the statistical probability that there is no significant difference between the mean insight scores of any two given viewing conditions. *We see here that VC is much better than NVNC (statistically significant at better than the 0.001 level), and that NVC is marginally better than NVNC.*

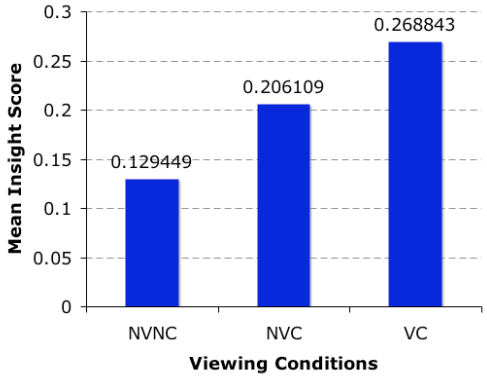


Figure 74: Comparison of mean insight score by viewing condition.

Table 17: Pairwise probabilities of no significant difference between the mean insight scores by viewing condition.

Between Conditions		Difference of Mean Insight scores	P-value
NVC	NVNC	0.076657	0.0727
NVC	VC	-0.062737	0.1643
NVNC	VC	-0.139393	**0.0004**

Ordering effects: One other question to examine was the effect of the order of viewing conditions on insight score. For example: Did participants score higher when they saw the visualization first? Since the experiment was not designed to quantify the effects of ordering, the order of the viewing conditions is a measured quantity, not an experimental factor.

The statistical consultants created a new *ordering* variable representing the order in which subject faced the treatment combinations. We constructed a hypothesis test to determine whether there were any significant effects of ordering on score. The result was marginally significant with $p = 0.0865$. A pairwise examination of the differences between mean scores from different orderings shows that the insight score was somewhat lower for the first treatment combination my subjects faced regardless of what it was. However the difference is only marginally significant.

To determine whether these results are true regardless of the scenario and condition used, we tested the interactions between the ordering variable and the scenario and condition variables. These results were extremely insignificant ($p = 0.5952$ and $p = 0.8833$ respectively). Thus the insight scores in each scenario or viewing condition were not significantly affected by the order they appeared.

I believe it is reasonable to expect people to do worse on an unfamiliar task on the first repetition than on later ones, and it is reassuring that they did so regardless of which treatment it was. Therefore I conclude that the ordering of treatment combinations did not adversely affect my interpretation of the insight scores.

6.3.6.3 Qualitative Findings

The shortcomings of the quantitative study lend increased importance to the qualitative findings the experiment yielded. Users reported the most satisfying experience from the VC, VNC, NVC, and NVNC conditions in that order. Interestingly though, when asked to rank the conditions from most to least insight gained, they frequently reversed the VNC and NVC conditions giving a ranking of: VC, NVC, VNC, NVNC (Figure 75). Many users had trouble relating the two visualization windows in the VNC condition, and I conclude that when forced to choose between visual presentation and correlated data, users prefer to look at the visualization, but may actually perform better with the correlated text data. The mean performance data appears as Figure 76. I made notes on user's reactions to the individual implementations of each condition. I have recorded some of the salient notes in Table 18.

Each of the four conditions has advantages and disadvantages. From the point of view of my study, the overall advantages and disadvantages of the visualization and correlation reported to and observed by me are summarized in Table 19.

HoNe's correlated visualization enables novice users with little experience to perform nearly as well and, in a few cases, better than highly experienced system administrators. I divided the test subjects into two groups based on experience level. Those who had worked as a professional system administrator for more than 12 months I considered *experienced* (12 of 27 subjects). The remainder (15 of 27 subjects) I considered *novices*.

I asked the statistical consultants whether experienced subjects scored better than novices, and they found that they did with marginal significance ($df = 1$, $SS = 0.059138$, $MS = 0.059138$, $F = 3.82$, p-value $= 0.0565$). The insight score for the experienced subjects was 0.054699 higher than for novices, with standard error 0.027997, $t = 1.95$, and p-value $= 0.0565$ (recalling that a t-test and an F-test will give the same results if the F-test has only 1 df in the numerator).

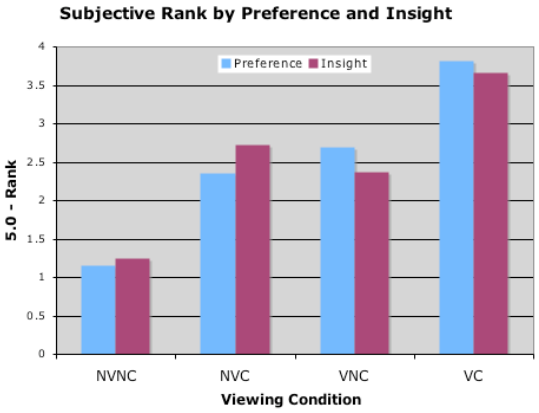


Figure 75: Mean rank of conditions by preference and insight gained.

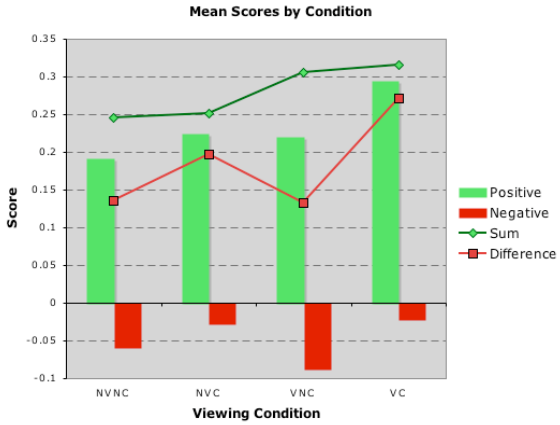


Figure 76: Mean positive and negative scores for each condition

I also asked the statistical consultants whether the experts or novices improved their scores more under either the NVC or VC conditions. The consultants approached the problem by testing whether the experienced subjects did better on a different scenario and/or condition than the novice subjects (e.g., testing experience*scenario interaction and experience*condition interaction). The result showed no significant difference in improvements (p-values 0.4880 and 0.7868). Thus, experience level does not influence improvement in score. Experts should already be proficient in forensic analysis so one would expect diminishing returns given greater user expertise. However, HoNe helped experts as much as it did neophytes showing that it offers valuable new insight.

Table 18: Advantages and disadvantages of each condition as reported by users

	Visualized	**Nonvisualized**
Correlated	Condition: **VC** + Users preferred this condition and said that it provided the most insight + On the average, users achieved the highest positive scores and least negative scores under this condition than under any other. – The slow performance and occasional bugs of the visualization caused problems for some users and may have affected the results.	Condition: **NVC** + Many users commented that they got better results using this condition than they did with the Visualized-Noncorrelated condition. + Users found that grepping out all the packet lines and just looking at the connection lines gave them a good overview of the file. – The correlated data provided a start and end timestamp for each connection, but a start time and a duration would have served them better. – Users most missed the visible representation of connection duration in this and the other text view.
Noncorrelated	Condition: **VNC** – Users found it very hard to correlate separate netstat and tcpdump windows, especially because the netstat timestamps were relative while tcpdump's were absolute. – Users would often remain in one screen or the other. – When they saw connections to odd ports in the tcpdump window, users assumed that the machine was listening on these ports even though the packet data would have shown them that the host reset the connection immediately.	Condition: **NVNC** + A few experienced users who were comfortable with grep, etc. could locate intrusions as fast with this condition as with a visualization. – Users frequently reported disliking this condition and found it hardest to understand. – I noticed that the most inexperienced users would dwell for long times in small parts of the data files or stay almost exclusively in one file. – Even experienced users who were accustomed to this kind of data made errors when trying to interpret the raw netstat data because of misleading information they found in it.

Table 19: Advantages and Disadvantages of various types of presentation

	Advantages	Disadvantages
Correlation	+ All important information is located in a single file or window. + Subjects said having the process name next to the connection data helped them diagnose connections.	– The correlated data necessarily omits some information thought to be less important. If this information is required, the user must correlate it in mentally.
Visualization	+ Attack patterns are immediately apparent—users can even differentiate between attackers by the visual "thumbprint" of the tools they use to attack. + Connection duration is immediately apparent. + Nonsequential analysis is possible. + It is easy to compare behaviors visually even when they are separated by many hours or days. + Users could make important discoveries by examining prominent visual features without knowing in advance what to look for. + The majority of subjects made their discoveries much quicker in visual conditions than in textual ones. + Subjects were, in general, more certain of their diagnoses when they arrived at them visually. + Subjects felt as if they understood what was really happening in the data better when it was presented to them visually. + The visual overview provides an immediate sense of context.	– A visualization immediately draws the user's attention to high-density events. Subtle but important events may hide in the clouds created by the visualization. – Users seemed less systematic when exploring under visual conditions. They skipped small features and browsed the data nonsequentially. – If there was an apparent contradiction between text and visual data, subjects seemed biased toward visual. – Some users stopped exploring the visualization after they had discovered what they thought they were supposed to find. They ignored the rest of the data. – When the overview covers a very large amount of data, visual compression results in a great degree of marker overlap hiding potentially important features. – Removing marker overlap is hazardous because attackers could more easily use disproportionately large attacks as camouflage. – Nearly half of all subjects never looked at the detailed text data provided for machines and connections. – The visualization program ran much more slower than text processing tools.
Text	+ Users who know what they are looking for and who are comfortable using regular expressions find text data more efficient. + No details are hidden from the user.	– Users often reported feeling "lost in the data." – Too much information is provided. – Greater expertise is required to efficiently manipulate text. – Inexperienced users covered only a small part of the data.

6.4 Overall Findings

It was my observation that users preferred the correlated visualization and felt they got the most insight from it of any of the conditions. I have demonstrated that the VC treatment resulted in better scores than the NVNC (control) at better than the 0.01 level of significance, and that the NVC treatment resulted in better scores than NVNC at the 0.1 level of significance. Additionally, I showed a marginally significant result showing that VC was better than NVC. From these findings, I infer that

my visual correlation of packets to processes does help administrators perform certain intrusion detection tasks better than text data alone could.

There were a large number of unsolicited positive comments about the VC condition and the visualization tool even though I did not tell the users that I was its creator unless they asked. Users both preferred the VC condition to all others and said that they received the most insight from it. I believe that users would have performed even better if not for some implementation bugs in the visualization and its slower than optimal responsiveness.

The NVC condition provided the critical element of packet-process correlation without providing a means of visualizing the data. The correlated data was much more compact than the noncorrelated and there was a single data file so users did not get lost nearly as easily as in the NVNC condition. Users who had high skills with text manipulation tools were often very adept at diagnosing problems in the NVC data. TCP conversation reconstruction (separating out individual pair-wise conversations from a larger set of many simultaneous conversations) was part of the correlation process that users mentioned was helpful. Although this is not a new contribution of the work, it made reading packet traces easier for humans. A typical activity I saw users perform with the NVC data was to mentally subtract the start and end times looking for long sessions. The correlated data would have been much more useful to humans if I had put a start time and duration instead of start and end times. In fact, users mentioned many times when using text data that they were looking for long sessions. Those who had performed a run under a visual condition previously often said that the visual indication of connection duration was one of the visualization's features they missed most.

The VNC condition turned out to be very troublesome both to users and to me. It seems that users expected a visualization to present only one window with all the data on it. Indeed, this is a goal of information visualization, and the users chafed at having two similar visualizations that told them slightly different aspects of the same data. Some of the confusion users had was due to the similar appearance but different meaning of the VC and VNC conditions. A better approach might have been to choose two separate visualizations, one tailored to displaying tcpdump data, and another tuned to show netstat data visually. In any case, it was more difficult than expected to test visualization and packet-process correlation as truly independent concepts.

Although some more experienced users did very well with the NVNC condition, it was the least liked and resulted in the lowest scores on average. Most users found this condition confusing and error-fraught. The most interesting phenomenon I noticed was how novice users approached the staggering amount of data this condition presented. Many times novices would start in one file and painstakingly try to understand each packet or connection attempt. Several novice users never looked at more than the first 1% of the packet dump data during the whole 15-minute run. Often, I would encourage these users to look at other data, but many novices seemed to be unable to draw high-level, evidence-based conclusions, preferring instead to interpret small findings deep in the details.

The user's reactions highlight an important general function of visualization, especially for novice users: visualizations present information compactly, allowing users to think about the data globally. In fact, both visualization and packet-process correlation have the effect of compacting and simplifying the data for human perception, although the effect of visualization seems to be more pronounced. In the future, other studies could be conducted in the user's actual work environments to see whether these conclusions hold true under more realistic conditions.

CHAPTER 7 CONCLUSION

Based on my study, interviews, and the literature, I believe I have demonstrated that correlating packets to the processes responsible for them and visualizing this correlation benefits system administrators. In this section, I will discuss my suggestions for follow-on work that may be done with this technology and present the contributions of the work to date.

7.1 Suggestions for future work

Conducting a lengthier study of several months duration where packet-process correlated data was collected from production systems would be very beneficial. However, the kernel modifications are not sufficiently stable to conduct such tests at this time. Thus, I suggest devising a stable version of these modifications based on a newer kernel. The correlated data could be collected from honeynets at first, backup servers next, and full production systems in the long run, once stability was more certain. Data collected from this study could be used to validate the findings of my study under more realistic conditions.

One possible route toward stability would be to influence the kernel community to adopt designs where this correlation is possible without drastic kernel modifications. This would open up the possibility of packet-process correlation to a much wider audience of users. Relaxing the strict separation of concerns between layers in the TCP/IP networking model would benefit other efforts as well such as the Quality of Service (QOS) community.

Of course, a production system may produce several orders of magnitude more data than used in the study. Modifications to the visualization and database will be needed to ensure that the visualization is responsive with large amounts of data. Similarly, users may wish to monitor numerous hosts simultaneously, so some attention needs to go toward increasing the scalability of the visualization so that more can be seen at a time.

Graphical aggregation of host markers by Classless Inter-Domain Routing (CIDR) blocks (as is done in Estan's Autofocus[45]) may also be a way to make HoNe more scalable. Such aggregations should only be done for unmonitored hosts where the process names are unknown. Level-of-detail and focus+context techniques could reduce the size of nonfocal markers to increase the display capacity.

Large numbers of connection lines may also be aggregated into a single line in some cases. Some of the users I interviewed operate web servers where thousands of connections per minute are normal. Automatic aggregation of connections to such servers from unmonitored hosts may be a way to reduce the visual clutter on the screen and increase the display's capacity. However, some study will be required to determine when to automatically aggregate and when to refrain from doing so. Large numbers of crossed lines can be a source of confusion as well. HoNe does not currently attempt to reduce the number of crossings, although heuristics such as the Iterated Barycenter method[78] may prove effective in a future version.

To move toward a complete, decentralized monitoring solution, HoNe will need infrastructural support for gathering correlated data files from remote hosts for visualization. The impact of secure connections between monitored hosts, aggregators, and visualization stations in a decentralized deployment is a worthy topic for future work. Beyond the basic scalability issues, communications among decentralized components may confound a user's perception of events on the monitored machines. Further research is required to investigate these scalability and decentralization issues.

TCP accounts for a little more than two-thirds of the traffic on the Internet. Other protocols such as UDP and ICMP provide other important sources of information. Sometimes packets of these protocols

are generated by the kernel itself rather than by a process, but tracking them would be very useful nonetheless. Thus, I recommend investigating the benefits of adding these common but connectionless protocols to future versions of the correlation engine.

Users greatly benefited from the filtering capabilities HoNe provides, but more are needed. Many more kinds of filters are conceivable and may be useful. However, the filter definition window is nearly as hard to use as the underlying SQL. Thus, I recommend adding more filtering to future versions, while simplifying the filter specification process to reduce the amount of training needed to become proficient in the use of filters.

The users in the HoNe pilot study asked for several expansions of the "more information" feature that I was unable to supply. Some users wanted to be able to receive up-to-date information on what malicious programs are using a given port number to communicate. This requires the availability of a reliable list of trojaned ports. I could not find such a list, but this information may be useful to accurate diagnosis of the monitored machines' conditions.

Another future expansion of the *More Information* pane would be to show what files each communicating process had open over its lifetime when it is double-clicked on. Collecting this information would require a tremendous amount of monitoring and storage unless some heuristic were discovered. Similarly, the kernel modifications could make use of the existing process accounting subsystems to help with diagnosing suspicious processes.

Monitoring of inter-process communication within the monitored machine would also be a helpful area for future work. For example, during an SQL injection attack, a web server might be the vector where the malicious code came into the system, but the database activities following the attack would be what appeared as suspicious. Thus the attack vector would be hidden unless there was a means of showing that the web server had instructed the database to execute the malicious code it received. Tracking inter-process communications would make the attack vector visible.

Finally, while this research focused on the benefits of visualized packet-process correlation, it would be instructive to determine the utility of this correlation for automated tools such as Intrusion Prevention Systems (IPS). Enabling IPS and firewalls to make decisions about network flows based on what process terminates the flow would provide a great advantage over current systems that are agnostic of process activity.

Not to dwell entirely on HoNe, future work with the high-density network visualization (particularly the root polar network pixel map) is worth continued study. One important future direction is to perform usability studies on polar vs. Cartesian coordinates to understand the cognitive implications of plot layout. Particularly, it would be good to know whether users can find particular addresses and determine a host's trust level on polar plots as quickly and accurately as they can on Cartesian plots. Another area for usability studies is to determine how many markers a user can comprehend on a single plot. The root polar plotting prototype works well for plots of 100,000 or more markers, but that does not mean that the plot is usable. Further studies are needed now that the technical groundwork has been laid.

7.2 Contributions

My research has advanced the science of computer security in several significant ways. In this section, I would like to recapitulate what I believe are the most important contributions of my work.

- I have interviewed system administrators and identified areas where HCI research could improve their tools.

- I have identified the host/network divide, shown its causes, and examined its effects on computer security both technologically and cognitively.
- With HoNe, I have bridged the technical aspects of the host/network divide and laid the groundwork for bridging the cognitive aspects as well.
- I have created a visualization of packet-process correlation, making it possible for humans to make better diagnoses about the nature of connections.
- I have demonstrated the advantages of packet to process correlation via quantitative usability evaluations.
- I have created a new source of correlated data that will be useful to automated security monitoring tools as well as humans.
- I have generated new tools for system administrators via participatory design[10] and performed usability evaluations to quantify their utility.

7.3 Impact

HoNe has demonstrated how helpful packet-process correlation and visualization are for detecting and diagnosing potentially malicious activity on computers. I have also shown that the layered model, while effective in many ways, has problems caused by lack of visibility across software layers within the kernel. As Cantrill[79] noted, the main problem with layered systems of today is a profound lack of software observability. The only way to see what software is doing, especially system software, is to modify it. This is what I have done with HoNe: I have modified the kernel to gain visibility into how packets relate to processes, and I have created a visualization of the information I have gathered. Since lack of observability is a huge problem in today's system software, I expect to see efforts such as DTrace[33] gain broader acceptance and become available on more platforms. But these programs only dump more text at the user. What I hope HoNe will show is how important it is to present computer security data to users in the way they can process it most rapidly, via visualization.

Much work remains to be done for system administrators, some of which I have outlined in the previous section. However, it is my hope that HoNe will lay the groundwork for a positive change in the way security professionals go about their work. If kernel designers accept my conclusions and incorporate greater observability of the packet-process relationship into their work, much better security monitoring can be done in the future than is possible today. System administrators will be able to interrogate their systems for security problems more directly and then visualize the results. This kind of progress will make it much harder for malicious persons to hide their activities, making the entire Internet safer for its users.

But in the design of tools for system administrators, I hope that the success of my work can demonstrate how important it is to talk to users and involve them as co-designers in any work that purports to meet a need. Tools can be more or less helpful depending on how well they adapt to their user's needs. The important thing is to find out what the users need and design tools to fit the need. HoNe is one such tool—may many others follow it.

REFERENCES

[1] Sun Microsystems, http://www.sun.com. Last accessed July 2006.

[2] Fink, G. A., Duggirala, V., Correa, R., and North, C. "Bridging the Host-Network Divide: Survey, Taxonomy, and Solution." To appear in *Proceedings of the USENIX conference on Large Information Systems Administration* (LISA), 2006.

[3] Rusling, D. A., The Linux Kernel, document version 0.8-3. http://www.tldp.org/LDP/tlk/tlk.html, 1999. Last accessed July 2006.

[4] Netfilter, http://www.netfilter.org/documentation/. Last accessed July 2006.

[5] RFC-791, Internet Protocol. Request for Comments 791, September, 1981, http://www.ietf.org/rfc/rfc0791.txt, (See also: MIL-STD-1777).

[6] RFC-793, Transmission Control Protocol. Request for Comments 793, September, 1981, http://www.ietf.org/rfc/rfc0793.txt, (See also: MIL-STD-1778).

[7] CS-5764 Information Visualization at Virginia Polytechnic Institute and State University, Fall 2002, http://infovis.cs.vt.edu/cs5764/. Last accessed July 2006.

[8] Rosson, M. B. and Carroll, J. M., *HCI Models, Theories, and Frameworks, Toward a Multidisciplinary Science*, San Francisco, Morgan-Kaufmann, 2003

[9] Rosson, M. B. and Carroll, J. M., *Usability Engineering: Scenario-based Design of Human-Computer Interaction*, San Francisco, Morgan-Kaufmann, 2002.

[10] Participatory Design, http://en.wikipedia.org/wiki/Participatory_design. Last accessed July, 2006.

[11] Yurcik, W., Meng, X. and Kiyanclar, N., NVisionCC: a visualization framework for high performance cluster security. In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, (Washington DC, USA, 2004), 133-137.

[12] Wickens, C.D., Sandry, D.L. and Vidulich, M. Compatibility and Resource Competition between Modalities of Input, Central Processing, and Output. Human Factors, 25 (2). 227-248.

[13] Tripwire. Tripwire, Inc. -- The Integrity Assurance Company http://www.tripwire.com/, Tripwire, Inc., 2003.

[14] PortSentry. Debian GNU/Linux -- portsentry, http://packages.debian.org/unstable/net/portsentry.html, The Debian Project (www.debian.org), 2003.

[15] MacGuire, S. and Croteau, R.-A. Big Brother, BB4 Technologies Inc, part of Quest Software Inc., 2003, http://www.quest.com/bigbrother/. Last accessed July 2006.

[16] Russell, R. ipchains: Linux IP Firewalling Chains, 2000, Linux ipchains is a rewrite of the Linux IPv4 firewalling code (which was mainly stolen from BSD) and a rewrite of ipfwadm , which was a rewrite of BSD's ipfw , I believe. It is required to administer the IP packet filters in Linux kernel versions 2.1.102 and above.

[17] Zone Alarm Pro, Zone Labs, Inc., San Francisco, CA 94107, USA, 2003, http://www.zonelabs.com/. Last accessed August, 2005.

[18] Arce, I. and Levy, E. An analysis of the slapper worm. IEEE Security & Privacy Magazine, 1 (1). 82-87.

[19] Ptacek, T.H. and Newsham, T.N. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, Secure Networks, Inc., 1998, 1-63.

[20] MyDoom Wikipedia entry, http://en.wikipedia.org/wiki/Mydoom. Last accessed July 2006.

[21] Treemaps, www.cs.umd.edu/hcil/treemap/. Last accessed July 2006.

[22] Fink, G. A., Muessig, P., and North, C. Visual correlation of host processes and network traffic. In Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security. IEEE, October 2005.

[23] Foundstone, Inc.'s free forensic tools, http://www.foundstone.com/resources/freetools.htm. Last accessed July 2006.

[24] Abell, V. Purdue University, Lsof Unix utility, http://freshmeat.net/projects/lsof/. Last accessed July 2006.

[25] "Information Visualization," *Readings in information visualization: Using vision to think*, Card, S.K., Mackinlay, J.D. and Shneiderman, B. eds., Morgan Kaufmann Publishers, San Francisco, Calif., pp.1-34, 1999.

[26] Center for Internet Security, http://www.cisecurity.org/. Last accessed July 2006.

[27] HP OpenView, http://www.managementsoftware.hp.com/. Last accessed July 2006.

[28] Nagios, http://www.nagios.org/. Last accessed July 2006.

[29] What's Up Gold, http://www.extralan.co.uk/products/Diagnostic-Tools/Ipswitch/Whats-Up-Gold.htm. Last accessed July 2006.

[30] eHealth, a network management tool owned by Computer Associates, Inc., http://www.concord.com/products/network_mgt.shtml. Last accessed July 2006.

[31] Orcallator, http://www.orcaware.com/orca/docs/orcallator.html. Last accessed July 2006.

[32] Visible Computer Applet, http://www.di.ufpe.br/~jhcf/vc/index_applet.html. Last accessed July 2006.

[33] Cantrill, B., Shapiro, M., and Leventhal, A. 2004. Dynamic instrumentation of production systems. Proceedings of the 2004 Usenix Annual Technical Conference.

[34] Kiran Lakkaraju, William Yurcik, and Adam J. Lee, "NVisionIP: netflow visualizations of system state for security situational awareness." Proceedings of VizSEC 2004, ACM Press, New York, NY, USA, pp. 65-72, October, 2004.

[35] Gkrellm, http://www.gkrellm.net/. Last accessed July 2006.

[36] Bosch, R., Stolte, C., Tang, D., Gerth, J., Rosenblum, M. and Honrahan, P. Rivet: A Flexible Environment for Computer Systems Visualization Computer Graphics, 2000, 68-73. See also "Rivet: The Visible Computer," http://graphics.stanford.edu/projects/rivet/. Last accessed July 2006.

[37] SGI's Performance Co-Pilot (PCP), http://oss.sgi.com/projects/pcp/. Last accessed July 2006.

[38] Polaris database visualization suite, http://graphics.stanford.edu/projects/polaris/. Last accessed July 2006.

[39] Richard Bejtlich, The Tao of Network Security Monitoring Beyond Intrusion Detection. Addison-Wesley, 2004.

[40] Tcpdump, http://www.tcpdump.org/. Last accessed July 2006.

[41] Ethereal, http://www.ethereal.com. Last accessed July 2006.

[42] Sguil, http://sguil.sourceforge.net/. Last accessed July 2006.

[43] Snort IDS, http://www.snort.org. Last accessed July 2006.

[44] Multi-Router Traffic Grapher (MRTG), http://oss.oetiker.ch/mrtg/index.en.html. Last accessed July 2006.

[45] Estan, C., Savage, S., and Varghese, G. "Automatically inferring patterns of resource consumption in network traffic," Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, ACM Press, New York, NY, USA, pp. 137-148, 2003.

[46] Cao, J., Cleveland, W.S. and Sun, D.X., S-Net: A Software System for Analyzing Packet Header Databases. In Passive and Active Measurement Workshop Proceedings, (Fort Collins, Colorado, USA, 2002), 34-44.

[47] Yin, X., Yurcik, W., Treaster, M., Li, Y. and Lakkaraju, K., VisFlowConnect: netflow visualizations of link relationships for security situational awareness. In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, (Washington DC, USA, 2004), ACM Press, New York, NY, USA, 26-34.

[48] Hyogon Kim, Inhye Kang, and Saewoong Bahk, "Real-time visualization of network attacks on high-speed links," IEEE Network, IEEE Press, Washington, DC, pp. 30-39, 2004.

[49] Lau, S. The Spinning Cube of Potential Doom. Communications of the ACM, 47 (6). 25-26.

[50] Secure Decision's Secure Scope, http://www.SecureDecisions.com/. Last accessed July 2006.

[51] Keim, D. A. Designing pixel-oriented visualization techniques: theory and applications. IEEE Transactions on Visualization and Computer Graphics, 6(1):59–78, 2000.

[52] MacKenzie,I.S., and Buxton,W. (1992). Extending Fitts' law to two-dimensional tasks. *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems*. New York:ACM.

[53] Ball R. G., Fink, G. A., Rathi A., Shah S., and North, C. "Home-centric visualization of network traffic for security administration." In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, International Conference on Information Visualization (IV2000), pages 55–64. ACM, ACM Press, 2004.

[54] Fink, G.A.; North, C. "Root polar layout of Internet address data for security administration." *In Proceedings of the IEEE Workshop on Visualization for Computer Security 2005* (VizSEC 05), pp. 55- 64, 26 Oct. 2005

[55] Bernardino, A., and Santos-Victor, J. Correlation based vergence control using log-polar images, 1996.

[56] Furnas, G. W. 1986. Generalized fisheye views. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Boston, Massachusetts, United States, April 13 - 17, 1986). M. Mantei and P. Orbeton, Eds. CHI '86. ACM Press, New York, NY, 16-23. DOI= http://doi.acm.org/10.1145/22627.22342

[57] Ben Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, CS-TR-3665, Department of Computer Science, University of Maryland, Human-Computer Interaction Laboratory, and Institute for Systems Research, 1996.

[58] Keim, D. A. and Herrmann, A. "The gridfit algorithm: An efficient and effective approach to visualizing large amounts of spatial data." In *Proceedings of the Conference on Visualization '98*, pages 181–188. IEEE Visualization, 1998.

[59] Nowell, L., Hetzler, E., and Tanasse, T. "Change blindness in information visualization: A case study." In *Proceedings of 2001 Information Visualization*, pages 15–22. IEEE Computer Society, 2001.

[60] Tversky B. "Distortions in cognitive maps." *Geoforum*, 23(2):131–138, 1992.

[61] Lippmann, R., Haines, J., Fried, D. J., Korba, J., and Das, K. "The 1999 DARPA Off-line Intrusion Detection Evaluation." *Computer Networks*, 34(4):579–595, October 2000.

[62] Graphviz, http://www.graphviz.org/. Last accessed July 2006.

[63] Ghostscript, http://www.ghostscript.com/. Last accessed July 2006.

[64] Microsoft .Net Framework, http://www.microsoft.com/net/. Last accessed July 2006.

[65] WinPCap: The Windows Packet Capture Library. http://www.winpcap.org/, Last accessed August, 2005.

[66] IPHLPAPI, http://windowssdk.msdn.microsoft.com/en-us/library/ms691199.aspx, Last accessed July 2006.

[67] AllocateAndGetTcpExTableFromStack, http://windowssdk.msdn.microsoft.com/en-us/library/ms691199.aspx. Last accessed July 2006.

[68] AllocateAndGetUdpExTableFromStack, http://windowssdk.msdn.microsoft.com/en-us/library/ms690716.aspx. Last accessed July 2006.

[69] Li, Q., North, C. "Empirical Comparison of Dynamic Query Sliders and Brushing Histograms." In *Proceedings of IEEE Symposium on Information Visualization 2003*, pp. 147-154, 2003.

[70] OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection, Hubert Zimmermann, IEEE Transactions on Communications, vol. 28, no. 4, April 1980, pp. 425 - 432.

[71] RFC 1122, "Requirements for Internet Hosts -- Communication Layers." http://tools.ietf.org/html?rfc=1122. Last accessed July 2006.

[72] SQLite 3, http://www.sqlite.org/. Last accessed July 2006.

[73] Stevens, W.R. TCP/IP Illustrated. Addison-Wesley, Reading, MA, US, 1994.

[74] North, C., "Toward Measuring Visualization Insight." In IEEE Computer Graphics and Applications, vol. 26, no. 3, pp. 6-9, 2006.

[75] User Mode Linux, http://user-mode-linux.sourceforge.net/. Last accessed July 2006.

[76] Adore kernel-level rootkit, http://www.packetstormsecurity.org/groups/teso/. Last accessed July 2006.

[77] LRK5 kernel-level rootkit, http://packetstormsecurity.org/UNIX/penetration/rootkits/lrk5.src.tar.gz. Last accessed July 2006.

[78] Junger, M. and Munzel, P. "2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms." In *Journal of Graph Algorithms and Applications,* vol. 1, no. 1, pp. 1–25, 1997.

[79] Cantrill, B. 2006. Hidden in plain sight. *Queue* 4, 1 (Feb. 2006), 26-36. DOI= http://doi.acm.org/10.1145/1117389.1117401

APPENDICES

Appendix A: Usability Evaluation Protocol

Visual Packet-Process Correlation
Interview Protocol

Setup Prior to each Experimental Run (10 min)

1. Assign order of conditions and scenarios according to the subject number.
2. Load the tcpdump data sets for the non-vis conditions into separate ethereal windows ahead of time since this takes lots of time.
3. Prepare two copies of the IRB informed consent form for this research.
4. Print a copy of the Interview Form to take notes on.
5. Ensure that all data and program files are available.
6. Erase any extraneous data files that the user has created.
7. Ensure Internet connectivity is working. Set up a connection to circlekcluster to keep the DHCP lease alive.
8. Start X-Windows and the HoNe program.
9. Load the sample scenario.
10. Ensure that only the standard set of HoNe filters is installed.

Do not discuss the purpose of the experiment or my work until after the experiment is complete.

Table 20: Assignment of conditions and scenarios to subjects.

Conditions: 1=NVNC, 2=NVC, 3=VNC, 4=VC

Subj #	Condition, Scenario Pairing			
1	1, 1	2, 2	3, 3	4, 4
2	2, 1	3, 5	4, 3	1, 4
3	3, 1	4, 2	1, 3	2, 4
4	3, 1	4, 2	1, 3	2, 4
5	4, 1	1, 2	2, 3	3, 4
6	1, 2	2, 3	3, 4	4, 1
7	2, 2	3, 3	4, 4	1, 1
8	3, 5	4, 3	1, 4	2, 1
9	4, 2	1, 3	2, 4	3, 1
10	1, 3	2, 4	3, 1	4, 2
11	2, 3	3, 4	4, 1	1, 2
12	3, 3	4, 4	1, 1	2, 2
13	4, 3	1, 4	2, 1	3, 5
14	1, 4	2, 1	3, 5	4, 3
15	2, 4	3, 1	4, 2	1, 3
16	3, 4	4, 1	1, 2	2, 3
17	4, 4	1, 1	2, 2	3, 3
18	4, 1	3, 5	2, 3	1, 4
19	1, 1	4, 2	3, 3	2, 4
20	2, 1	1, 2	4, 3	3, 4
21	3, 1	2, 2	1, 3	4, 4
22	4, 2	3, 3	2, 4	1, 1
23	1, 2	4, 3	3, 4	2, 1
24	2, 2	1, 3	4, 4	3, 1
25	3, 5	2, 3	1, 4	4, 1
26	4, 3	3, 4	1, 2	2, 1
27	1, 3	4, 4	3, 1	2, 2

1.0 Front Matter (20 min)

1.1 Introduction

My name is <interviewer> and my research group is collecting information to determine what features of security awareness tools provide the most insight to users investigating potential intrusions. We are not evaluating you in any way; we are only trying to evaluate how well our tool works. Please be honest and don't be afraid to say negative things. All kinds of feedback are helpful.

None of your personal information (name, e-mail, etc.) will be released outside our research group to protect your anonymity. The only identifier we will use is your first name and your subject number. Participation is strictly voluntary, and you may refuse to answer any question or terminate the experiment at any time. The interview should last between 60 and 90 minutes.

Informed Consent: Give subject two copies of the IRB informed consent document, one to keep and another to sign and return.

I will show you four computer security data sets (or scenarios) under four different conditions. In two conditions you will use a visualization and in two you will use plain text data. Half of the scenarios will have correlated data where you will be able to see most everything you need from within a single window. The other half will have data from more than one window that you will have to correlate yourself. I will now show you examples of each kind of data we are using.

1.2 Training and Orientation

1. Show sample netstat data
2. Show sample lsof data
3. Show sample tcpdump data (show an example BPF filter)
4. Explain how information can be extracted from text files via grep
5. Show how to read the textual correlated data
6. Start HoNe and explain:
    a. The major screen areas (detailed view, time windows, more information, and filters)
    b. Detailed view:
        * Explain the areas of the detailed view (managed vs. unmanaged, enterprise vs. foreign)
        * Explain the meanings of the icons, lines, and colors
        * Show how to get more information on an object (double-click)
        * Show how to shrink or expand an area and how to close or open a box.
    c. Time windows:
        * Explain the levels of time
        * Explain the meaning of the histogram bars
        * Explain the meaning of the connection lines
        * Show how to select a time window
        * Show how to select an area to display in the detailed view
        * Show how to get more information about a connection
    d. Filters tab
        * Show how to add a filter
        * Show how to edit a filter
        * Show how to change filter actions and styles

* Show how to find and inverse-find using a filter

e. Explain the visible differences between viewing correlated and non-correlated data.

1.3 Scoring

I will score your success under four different conditions, and the better you score, the more chances you'll have to win. First prize will be $50 (available only to the top 20% scorers). Second prize is $25 (available to the scorers in the top 50%). There will also be two participation prizes of $15 each available to anyone who helps out. You may win zero or more prizes up to a maximum of $105. Scores will be calculated and cash prizes awarded after the conclusion of the study. You will not be told your score today. There will be about 30 participants total.

You will be scored on the number of insights you gain from the data set and the accuracy of those insights. Here are some categories of events you might identify in the data:

1. Unusual levels of activity during a certain time period.
2. Abnormal kinds of activity (given the definition of normal provided)
3. Incidents, events, or probable attacks.

Please diagnose whether the each event was benign or malicious. For malicious events, decide whether the attack actually penetrated the machine allowing the attacker access to the machine. Note: Misidentifying or misdiagnosing events will actually cost you points, so try to be as accurate as possible and not too paranoid.

1.4 Questions

Please "think-aloud" while you are evaluating each scenario. I will ask you the following questions during each scenario:

1. What network accessible services (apart from ssh) are running on the monitored machine?
2. Which (if any) ports besides port 22 in use during the monitoring period?
3. Describe what is happening in this scenario.
4. What are the advantages of looking at computer security data in this way?
5. What are the disadvantages of looking at computer security data in this way?

1.5 Beginning the Recorded Section

We must record audio of this experiment in order to gather data later. We will use the audio recordings to take notes from. We may transcribe certain things you say, but we will not attribute these things to you in any publication. Would you allow us to record this experiment? *<if "yes," start recording **and start the timer***>

Today is <date> at <time> and this is <interviewer> interviewing <subject's first name>. <Subject's first name>, you have agreed to let us record this interview for the purpose of taking notes later. We agree to preserve your anonymity outside our research group, not to release any personally identifiable information, and to anonymized any direct quotes from you. Are you comfortable with this arrangement? *<if "no," stop the recording and terminate the experiment>*

2.0 Background information (10 min)

- Have you had any experience administering computer systems professionally?
  - <if "yes", record how many>
  - <if "no" skip to question □.>
- How many user workstations (including lab machines and personal workstations) do you administer?

- How many server machines <machines that provide a service to more than a single user at a time> do you administer?
- What types of operating systems are installed on the systems you administrate?
- What level of education have you attained? Ex. High school, Associates, Bachelors, Masters, Doctoral.
- Have you earned any special computer-security related certifications? If so, what are they and how recently did you earn them?
- How familiar are you with the using netstat? (e.g., Expert, Power User, Competent User, Basic User, Unfamiliar)
- Have you ever used lsof? If so, describe your competence with the tool. (as above)
- Have you ever used tcpump (or ethereal, etc.)? If so, describe your competence with the tool. (as above)
- Have you ever used a visualization to monitor the security of a computer system? If so, which tools?
- Have you ever used a tool that correlated communications activity on the network to process activity on the monitored host? If so, which tools?
- Have you ever investigated a suspected computer intrusion? How often?
  - If so, have any of the incidents turned out to be actual intrusions?
  - How would you characterize your abilities to verify whether an incidents is an intrusion or not? (e.g., Expert, Knowledgeable, Average Competence for my profession, Limited Understanding, Clueless.).

3.0 Conduct of Experiment (60 min)

3.1 Scenarios 1-4 (15 min/scenario)

Condition 1:

1. Bring up at least one xterm
2. Load the scenario.netstat, scenario.lsof, and scenario.fulllog files into text editors
3. Load ethereal with scenario.tcpdump data
4. Explain that the subject may also use tcpdump with BPF filters to look at the tcpdump data
5. Explain that the subject may sift through data using any Unix program that does not destroy the original data files
6. Explain that the subject may ask for and receive help using any tool without penalty to his score.

Condition 2: Same as condition 1, but add the textual correlated data

Condition 3:

1. Bring up an instance of HoNe and load the netstat.dbfile data file into it
2. Bring up a second instance of HoNe and load the tcpdump.dbfile data file into it.
3. Explain that the subject may not use the scenario.* data files for this scenario.
4. Explain that the subject may ask for and receive help using the visualization without penalty to his score.

Condition 4: Same as condition 3 except there is only one instance of HoNe loaded with the iptab.dbfile data file.

3.2 Conclusion

- Please rank the conditions from most satisfying to least.
- Please rank them from most insight offered to least.
- Do you have any questions or comments? Thank you for your time! <*Pause the recording*>
- If you have any further questions or concerns about this study or if you would like to withdraw your participation prior to publication of our results, please feel free to contact me at <contact info> or write finkga@vt.edu. Thank you again for your help.

4.0 Post-Interview Recap (10 min)

After the subject leaves, record the salient points of the interview:

- How was this subject typical?
- How was he unique?
- How much data did he cover in each case?

Visual Packet-Process Correlation
Interview Data Sheet

Subject #: _____
Background

1. Years of experience administering computer systems professionally?
2. Number of user workstations administered?
3. Number of server machines administered?
4. Types of operating systems are installed on the systems administered?
5. Level of education have you attained?
6. Special computer-security related certifications/date?
7. Familiarity with netstat? (e.g., Expert, Power User, Competent User, Basic User, Unfamiliar)
8. Familiarity with lsof?
9. Familiarity with tcpump (or ethereal, etc.)?
10. Computer security visualizations used?
11. Communications/process correlation tools used?
12. Ever investigated a suspected computer intrusion? How often?
13. Any actual intrusions?
14. Characterization of abilities to verify an intrusion? (e.g., Expert, Knowledgeable, Average Competence for my profession, Limited Understanding, Clueless.).

Run 1: Condition: _____, Scenario: _____
Recording Cue: _____
Observations:

Overall Diagnosis:

Advantages of this method:

Disadvantages of this method:

Run 2: Condition: _____, Scenario: _____

Recording Cue: _____
Observations:

Overall Diagnosis:

Advantages of this method:

Disadvantages of this method:

Run 3: Condition: _____, Scenario: _____
Recording Cue: _____
Observations:

Overall Diagnosis:

Advantages of this method:

Disadvantages of this method:

Run 4: Condition: _____, Scenario: _____
Recording Cue: _____
Observations:

Overall Diagnosis:

Advantages of this method:

Disadvantages of this method:

Overall impression of the interview, tool, and experiment:

Rank by Preference Rank by Insight

Appendix B: Usability Evaluation Scenarios

Points for diagnosis as…			**Scenario #1: Non-hacked**	
MP	NM	NP	Event Description	Event ID
Malicious Non-Penetrating:				
-2	-1	3	SSH Dictionary from 199.77.146.106 (imcrypto2) 15 Mar 06 from 1545 to 1630.	S1NP01
-1	-1	3	SSH Dictionary from 80.55.79.245 (sb245.internetdsl.tpnet.pl) 15 Mar 06 from 1845 to 1855 (but may appear MP in some views)	S1NP02
-1	-1	2	SSH Dictionary from 72.66.186.3 (verizon.net) 15 Mar 2006 from 2141 to 2232 (but may appear MP in some views)	S1NP03
-2	-1	2	Misc. SSH Attempts from 61.62.46.105 (Taiwan, DSL), 80.113.103.2 (Netherlands), 131.188.154.27 (Germany), 201.5.203.205 (Brazil, Dial-up), 211.169.240.201 (Korea), and 216.75.14.141 (California)	S1NP04
-2	-1	2	Miscellaneous attacks on common ports 80, 135, 137, 139, 143, 445, 1026, 1027, 1434, 4000, 6000, 6101, 6112, 8080, 9544, 10000, 13701, 13722 by various attackers	S1NP05
Malicious Penetrating:				
3	-1	1	Apparent: All Conditions: There are a number of > 20.0 sec connections from 199.77.146.106	S1MP01
2	-1	-2	Apparent: All conditions: Long (56 sec in iptab, 138 sec in netstat/tcpdump) connection from 72.66.186.3 is actually me collecting the data, but since I had just done an SSH attack from that host it looks like an attacker got in. This was a mistake on my part, but reasonable for users to peg as MP.	S1MP02
1	1	1	Apparent: Condition 3 (tcpdump data): Long DNS "connection"	S1MP03
3	-2	-3	Apparent: Condition 4: Four "connections" from 80.55.79.245 appear in VC view to last over five minutes, but they are actually each 2 separate connections on the same port that the database builder has joined. That these are not long connections can be determined in the visualization only by looking at the packet dump	S1MP04
Non-features:				
-3	1	-2	Short connection to green 12:19, 15 Mar 06	S1NM01
-3	1	-1	Lots of DNS activity coincident with SSH DOS's from 199.77.146.106 (planned attacker)	S1NM02
-2	1	1	In netstat data, reference to mysql, root, etc. user names in an SSH attack	S1NM03
Extra Credit:				
2	2	2	Condition 4 Viewers who notice that the "long connections" from 80.55.79.245 are actually more than one connection each.	S1EC01

1	1	1	Condition 3 Viewers who notice that the long DNS "connection" to the campus DNS servers is really nothing but a series of UDP packets and not a connection at all.	S1EC02
1	1	1	Conditions 1 and 2 users who notice that the SSH Dictionary from 72.66.186.3 (verizon.net) 15 Mar 2006 from 2141 to 2232 was conducted by me	S1EC03
1	1	1	Finding out that 199.77.146.106 is associated with GA Tech	S1EC04

Maximum Possible Score: 29
Minimum Possible Score: -22

Points for diagnosis as…			**Scenario #2: Engineered Hack**	
MP	NM	NP	Event Description	Event ID
Malicious Non-Penetrating:				
-2	-1	2	One short and one long (33.9 sec) SSH connection from 68.107.98.210 (cable modem from Cox.net in Atlanta) at 00:35 and 00:47 respectively.	S2NP01
-2	-1	2	SSH Probe from 163.23.203.72 (Taiwan) at 0031 followed by SSH Dictionary attack from same IP from 05:46 to 08:09.	S2NP02
-2	-1	2	Misc. SSH connections from 222.82.32.135 (China), and 72.66.186.3 (Verizon, me)	S2NP03
-2	-1	2	Incoming SSH from 201.136.129.173 (Mexico City)	S2NP04
-1	0	1	Host 163.23.203.72 (Taiwan) sends strings of RST packets after close	S2NP05
Malicious Penetrating:				
2	-3	-1	SSH Dictionary attack from 199.77.146.106 (imcrypto2) from 10:45 to 11:15 as cover for main attack.	S2MP01
1	-3	-2	Long SSH connections from 199.77.146.106 (imcrypto2) for 37 sec, 804 sec	S2MP02
1	-2	-1	Apparent: Condition 4: Outgoing SSH from monitored host to 199.77.146.106 (imcrypto2). Appears to be a fragment of an earlier connection that somehow got separated.	S2MP03
2	1	1	Outgoing wget connection to 62.149.130.49 (aruba.it, Italy) on port 80 to download the rootkit.	S2MP04
1	-1	-1	Outgoing connections by process "./ntpd" and "" to 199.77.146.106:59001	S2MP05
1	-1	-1	Outgoing connection to 198.82.143.115:3440 (circlekcluster.cs.vt.edu)	S2MP06
1	-1	-1	Incoming connections from 198.82.143.115 and 199.77.146.106 on ports 6667 and 6668 indicates IRC bot named "./ntpd"	S2MP07
1	-1	-1	Long (7,312 sec) connection to monitored host port 6667.	S2MP08
1	-1	-1	./ntpd running on monitored host under usernames "service" and "root".	S2MP09

Non-features:

-2	1	-1	Logins from 128.173.54.105 (me picking up data)	S2NM01
-2	0	-1	Crond running as a service (not network accessible)	S2NM02
-1	1	0	In netstat data, reference to "[accept", etc. during an SSH attack	S2NM03
-1	0	0	In netstat data, init process owning sockets	S2NM04

Extra Credit:

2	2	2	ntpd trojan running out of /tmp/…/.bash/ntpd indicates phony exec.	S2EC01
1	1	1	wget and sshd: programs not shown in lsof probably due to file being truncated.	S2EC02
1	1	1	Finding out that 199.77.146.106 is associated with GA Tech	S2EC03
1	1	1	Able to tell the intrusion story partially	S2EC04
1	1	1	Able to tell the intrusion story accurately	S2EC05

Maximum Possible Score: 28
Minimum Possible Score: -28

Points for diagnosis as…			**Scenario #3: Non-hacked**	
MP	NM	NP	Event Description	Event ID

Malicious Non-Penetrating:

-2	-1	3	Small SSH scan from 72.11.128.138 (unknown, OC3 Networks?). 22 connections starting	S3NP01
-2	-1	3	SSH probe from 72.29.72.47 (server3.profitgate.com) at 16 Mar 2006 21:17:19 followed by full SSH Dictionary attack from same from 16 Mar 2006 23:03:14 to 23:25:03.	S3NP02
-2	-1	3	Misc. SSH Probes from 212.38.235.35 (Finland), 164.100.112.72 scdlr.mah.nic.in (India), and 61.177.251.2 (China). Assign 1 point per IP noticed up to 3.	S3NP03
-2	-1	3	Portscan from 222.190.210.187 (China) on 17 Mar 2006 at 13:24, 38 connections to 14 ports.	S3NP04

Malicious Penetrating:

none

Non-features:

-2	1	-1	Long SSH (34 sec) from 128.173.54.105 (green.cirt.vt.edu, me) to collect data.	S3NM01
-3	1	-1	Long DNS (22 min) "connection to 198.82.247.34 at 16 Mar 2006 23:03.	S3NM02
-3	1	1	Tcpdump network noise attacks.	S3NM03

Extra Credit:

1	1	1	Viewers who notice that the long DNS "connection" to the campus DNS servers is really nothing but a series of UDP packets and not a connection at all.	S3EC01

Maximum Possible Score: 16
Minimum Possible Score: -16

Points for diagnosis as…			**Scenario #4: External Hack**	
MP	NM	NP	Event Description	Event ID
Malicious Non-Penetrating:				
-3	-3	3	Misc. SSH probes and minor attacks from: 211.68.160.8, 219.142.1.118, 220.248.17.189, 219.134.241.136, 61.143.54.104 222.191.240.198 (all China), 128.173.54.113 (candi4.cirt.vt.edu), 193.252.7.123 (cable modem from France), 125.250.248.130 (Korea). One point per IP up to a max of 3.	S4NP01
-1	-1	1	SYN Attack on SSH port from 210.205.6.163 (Korea) – Looks like a long SSH connection from within the vis, but attacker only sends a SYN, while the rest of the 10 minutes the server sends SYN-ACKs to try to complete the handshake. User must look at the packets to know it is NP.	S4NP02
-3	-2	3	SSH Dictionary from 61.54.44.146 (China) on 3/26 from 04:38 to 04:56.	S4NP03
-3	-2	3	SSH Dictionary attack from 61.219.44.250 (Taiwan) on 3/27 from 02:04 to 02:18. Traffic has unusual shape starting with 4 sec connections increasing to 5 sec, and then backing off to fractions of seconds.	S4NP04
-2	-1	2	Small SSH Dictionary from 202.63.163.98 (India)	S4NP05
Malicious Penetrating:				
3	-3	-3	Long SSH (5 min) connection from 81.196.144.243 (statia13.moonlightclub.ro, Romania). Attacker logs in to account "user" right away indicating the password was guessed earlier.	S4MP01
2	-1	-2	Outgoing FTPs to 62.168.109.152 (creon.host.sk, Slovakia) evidently to download toolkit.	S4MP02
3	-3	-3	New services named "sshd:" start up and begin numerous outbound connections to a variety of hosts on server ports 6662, 6667, 7000, 7777, 8000, 8080, and 8888	S4MP03
3	-3	-3	Long SSH (11 min) from 81.196.144.240 (another Romanian host on the attacker's same subnet).	S4MP04
3	-1	-3	Outgoing SSH attempts on hosts 128.173.54.105 (green), 128.173.54.205 (homer.cs.vt.edu), and 128.173.54.203 (evostorm.cs.vt.edu). This happens during login from 81.196.144.240 (Romania). Connection to homer lasts two minutes, but most of it is dead space.	S4MP05
3	-1	-3	Really long (71,000 sec, nearly a day) connection from a trojaned sshd to 62.231.74.10 port 6667.	S4MP06

Non-features:

-3	1	-2	Several long SSH sessions from 128.173.54.105 (green.cirt.vt.edu, me).	S4NM01
-3	1	-2	Tcpdump noise attacks	S4NM02

Extra Credit:

2	2	2	Noticing that no new services are opened, thus privilege escalation has likely not occurred.	S4EC01
3	3	3	Able to tell all or part of the intrusion story (1-3 points)	S4EC02
2	2	2	Noting that 202.63.163.98 (India) and 61.219.44.250 (Taiwan) actually managed to login.	S4EC03
				S4EC04
				S4EC05

Maximum Possible Score: 35
Minimum Possible Score: -32

Points for diagnosis as…			**Scenario #5: Nonhacked (backup scenario)**	
MP	NM	NP	Event Description	Event ID

Malicious Non-Penetrating:

-3	-3	3	SSH Probes from 213.186.40.112 (France) on 3/21 at 16:09, and from 59.42.83.204 (China) at 17:08 and 17:20. On 3/22 from 212.92.14.82 (Hungary) at 05:13 and 05:24, 83.142.52.211 (UK) at 08:25, and 61.177.251.2 (China) at 14:04. One point per IP negative or positive up to 3.	S5NP01
-2	-2	2	Longish (36 sec) SSH attempt from 81.196.144.236 (statia6.moonlightclub.ro) trying to get in as a previously compromised user, "test" at 17:08 and again at 20:07.	S5NP02
-3	-2	3	SSH Dictionary from 211.137.77.150 (China) at 18:03.	S5NP03
-3	-2	3	SSH Probe from 213.193.230.2 (Belgium) at 20:11 followed by a full SSH Dictionary at 23:30 for 15 min.	S5NP04
-2	-1	2	Small SSH Dictionary from 61.177.251.2 (China) on 3/22 from 05:56 to 05:57 (8 connections)	S5NP05
				S5NP06

Malicious Penetrating:

3	-2	-2	Apparent: Conditions 3 and 4: SSH connection from 83.142.52.211 appears to last 85 seconds. Users who classify this as NP who also notice that it is not a complete connection gain 3 points. Other NP classifications lose 2.	S5MP01
3	-2	-2	Apparent: Conditions 3 and 4: 54 sec SSH connection from 211.137.77.150 (China) during the SSH attack actually is mostly waiting for the final FIN, but in the vis it looks like a substantive connection. User who notice its lack of substance in the packet trace gain 2 points, other NPs lose 2.	S5MP02

3	-2	-2	Apparent: Condition 3: 800 sec DNS connection (actually a series of queries that are close in time but have been grouped by the tool into one seemingly long session. Looks malicious.	S5MP03

Non-features:

-1	1	-1	Long SSH (215 sec) from 72.66.186.3 (Verizon, me) logging in.	S5NM01
-3	1	-2	Tcpdump junk	S5NM02
				S5NM03

Extra Credit:

1	1	1	TCPdump users who note a correlation between SSH attacks and long DNS	S5EC01
				S5EC02
				S5EC03

Maximum Possible Score:	25
Minimum Possible Score:	-23

Appendix C: Usability Evaluation Results

These are the results from the usability evaluation used in the analysis. The column definitions are:

1. **Run**: The experiment repetition number (assigned *ex post facto*). Note that only one subject was tested at a time and (with only two exceptions) all subjects' runs were done in a single sitting. Run number preserves the order the tests were conducted within subjects.
2. **Subj**: The subject unique identifier.
3. **Exp**: Subject experience level (0 = < 1 year, 1 = 1+ years of professional system administration experience).
4. **Cond**: Viewing condition text name.
5. **Vis**: Viewing condition: 1 = via visualization, 0 = text only.
6. **PPC**: Viewing condition: 1 = process-packet correlated, 0 = noncorrelated.
7. **Scn**: Scenario (dataset) number used.
8. **ScorePos**: Normalized points awarded for noticing and correctly diagnosing features of the dataset.
9. **ScoreNeg**: Normalized points deducted for noticing and incorrectly diagnosing features of the dataset.
10. **Insight**: ScorePos – ScoreNeg.

Run	Subj	Exp	Cond	vis	ppc	scn	ScorePos	ScoreNeg	Insight
1	1	1	NVNC	0	0	1	0.0000	0.1364	-0.1364
2	1	1	NVC	0	1	2	0.2143	0.1786	0.0357
3	1	1	VNC	1	0	3	0.0000	0.3125	-0.3125
4	1	1	VC	1	1	4	0.3143	0.0000	0.3143
5	2	0	NVC	0	1	1	0.0000	0.0000	0.0000
6	2	0	VNC	1	0	5	0.1600	0.0000	0.1600
7	2	0	VC	1	1	3	0.3750	0.0000	0.3750
8	2	0	NVNC	0	0	4	0.0857	0.0625	0.0232
9	3	0	VNC	1	0	1	0.1034	0.0455	0.0580
10	3	0	VC	1	1	2	0.1786	0.0000	0.1786
11	3	0	NVNC	0	0	3	0.1875	0.0000	0.1875
12	3	0	NVC	0	1	4	0.1429	0.0000	0.1429
13	4	0	VNC	1	0	1	0.0690	0.2273	-0.1583
14	4	0	VC	1	1	2	0.3571	0.0000	0.3571
15	4	0	NVNC	0	0	3	0.1250	0.1250	0.0000
16	4	0	NVC	0	1	4	0.1143	0.0313	0.0830
17	5	1	VC	1	1	1	0.2759	0.0909	0.1850
18	5	1	NVNC	0	0	2	0.2143	0.1071	0.1071
19	5	1	NVC	0	1	3	0.5625	0.0000	0.5625
20	5	1	VNC	1	0	4	0.2571	0.0000	0.2571
21	6	0	NVNC	0	0	2	0.0000	0.1071	-0.1071
22	6	0	NVC	0	1	3	0.4375	0.0000	0.4375
23	6	0	VNC	1	0	4	0.2571	0.0000	0.2571

24	6	0	VC	1	1	1	0.0000	0.0909	-0.0909
25	7	0	NVC	0	1	2	0.2500	0.0000	0.2500
26	7	0	VNC	1	0	3	0.5000	0.0000	0.5000
27	7	0	VC	1	1	4	0.4000	0.0313	0.3688
28	7	0	NVNC	0	0	1	0.1034	0.0000	0.1034
29	8	0	VNC	1	0	5	0.1200	0.1304	-0.0104
30	8	0	VC	1	1	3	0.3750	0.0000	0.3750
31	8	0	NVNC	0	0	4	0.0857	0.0000	0.0857
32	8	0	NVC	0	1	1	0.1034	0.0000	0.1034
33	9	1	VC	1	1	2	0.2143	0.0000	0.2143
34	9	1	NVNC	0	0	3	0.4375	0.0000	0.4375
35	9	1	NVC	0	1	4	0.0857	0.0000	0.0857
36	9	1	VNC	1	0	1	0.1724	0.1364	0.0361
37	10	0	NVNC	0	0	3	0.0625	0.1250	-0.0625
38	10	0	NVC	0	1	4	0.1714	0.0000	0.1714
39	10	0	VNC	1	0	1	0.0690	0.2273	-0.1583
40	10	0	VC	1	1	2	0.1786	0.0000	0.1786
41	11	1	NVC	0	1	3	0.1875	0.0000	0.1875
42	11	1	VNC	1	0	4	0.1143	0.0938	0.0205
43	11	1	VC	1	1	1	0.3793	0.0909	0.2884
44	11	1	NVNC	0	0	2	0.1071	0.0357	0.0714
45	12	0	VNC	1	0	3	0.2500	0.0000	0.2500
46	12	0	VC	1	1	4	0.1143	0.0000	0.1143
47	12	0	NVNC	0	0	1	0.0000	0.0000	0.0000
48	12	0	NVC	0	1	2	0.2500	0.0000	0.2500
49	13	0	VC	1	1	3	0.3750	0.0000	0.3750
50	13	0	NVNC	0	0	4	0.0857	0.0000	0.0857
51	13	0	NVC	0	1	1	0.1724	0.0000	0.1724
52	13	0	VNC	1	0	5	0.1200	0.0870	0.0330
53	14	1	NVNC	0	0	4	0.2571	0.0000	0.2571
54	14	1	NVC	0	1	1	0.1034	0.0000	0.1034
55	14	1	VNC	1	0	5	0.2400	0.1304	0.1096
56	14	1	VC	1	1	3	0.3750	0.0000	0.3750
57	15	0	NVC	0	1	4	0.2571	0.0938	0.1634
58	15	0	VNC	1	0	1	0.2759	0.2273	0.0486
59	15	0	VC	1	1	2	0.1429	0.0000	0.1429
60	15	0	NVNC	0	0	3	0.4375	0.0000	0.4375
61	16	0	VNC	1	0	4	0.0857	0.0938	-0.0080
62	16	0	VC	1	1	1	0.3448	0.1364	0.2085
63	16	0	NVNC	0	0	2	0.2857	0.1786	0.1071
64	16	0	NVC	0	1	3	0.3750	0.0000	0.3750
65	17	0	VC	1	1	4	0.1429	0.0000	0.1429
66	17	0	NVNC	0	0	1	0.1724	0.0909	0.0815
67	17	0	NVC	0	1	2	0.0000	0.0357	-0.0357
68	17	0	VNC	1	0	3	0.2500	0.0000	0.2500

69	18	0	VC	1	1	1	0.3103	0.0455	0.2649
70	18	0	VNC	1	0	5	0.1600	0.4783	-0.3183
71	18	0	NVC	0	1	3	0.4375	0.0000	0.4375
72	18	0	NVNC	0	0	4	0.2571	0.0000	0.2571
73	19	0	NVNC	0	0	1	0.2069	0.0000	0.2069
74	19	0	VC	1	1	2	0.3929	0.0000	0.3929
75	19	0	VNC	1	0	3	0.5000	0.0000	0.5000
76	19	0	NVC	0	1	4	0.2571	0.0000	0.2571
77	20	0	NVC	0	1	1	0.1034	0.0909	0.0125
78	20	0	NVNC	0	0	2	0.2143	0.0357	0.1786
79	20	0	VC	1	1	3	0.1875	0.0000	0.1875
80	20	0	VNC	1	0	4	0.0857	0.0938	-0.0080
81	21	1	VNC	1	0	1	0.3793	0.0455	0.3339
82	21	1	NVC	0	1	2	0.1429	0.0714	0.0714
83	21	1	NVNC	0	0	3	0.3750	0.0000	0.3750
84	21	1	VC	1	1	4	0.4857	0.0000	0.4857
85	22	1	VC	1	1	2	0.2857	0.0000	0.2857
86	22	1	VNC	1	0	3	0.3125	0.0000	0.3125
87	22	1	NVC	0	1	4	0.2571	0.0000	0.2571
88	22	1	NVNC	0	0	1	0.2069	0.3182	-0.1113
89	23	1	NVNC	0	0	2	0.0000	0.0714	-0.0714
90	23	1	VC	1	1	3	0.3750	0.0000	0.3750
91	23	1	VNC	1	0	4	0.2857	0.0000	0.2857
92	23	1	NVC	0	1	1	0.2759	0.0000	0.2759
93	24	1	NVC	0	1	2	0.2143	0.0000	0.2143
94	24	1	NVNC	0	0	3	0.3750	0.0625	0.3125
95	24	1	VC	1	1	4	0.4286	0.0000	0.4286
96	24	1	VNC	1	0	1	0.3793	0.0000	0.3793
97	25	1	VNC	1	0	5	0.3600	0.0435	0.3165
98	25	1	NVC	0	1	3	0.3125	0.1250	0.1875
99	25	1	NVNC	0	0	2	0.1786	0.0000	0.1786
100	25	1	VC	1	1	1	0.2759	0.0909	0.1850
101	26	1	VC	1	1	3	0.4375	0.0000	0.4375
102	26	1	VNC	1	0	4	0.4286	0.0000	0.4286
103	26	1	NVNC	0	0	2	0.4643	0.0000	0.4643
104	26	1	NVC	0	1	1	0.4828	0.0455	0.4373
105	27	1	NVNC	0	0	3	0.1875	0.0000	0.1875
106	27	1	VC	1	1	4	0.2000	0.0000	0.2000
107	27	1	VNC	1	0	1	0.0000	0.0000	0.0000
108	27	1	NVC	0	1	2	0.2143	0.0000	0.2143