Bridging the Host-Network Divide: Survey, Taxonomy, and Solution

Glenn A. Fink and Vyas Duggirala – Virginia Polytechnic Institute and State University
Ricardo Correa – University of Pennsylvania
Chris North – Virginia Polytechnic Institute and State University

ABSTRACT

This paper presents a new direction in security awareness tools for system administration-the Host-Network (HoNe) Visualizer. Our requirements for the HoNe Visualizer come from needs system administrators expressed in interviews, from reviewing the literature, and from conducting usability studies with prototypes. We present a tool taxonomy that serves as a framework for our literature review, and we use the taxonomy to show what is missing in the administrator's arsenal. Then we unveil our tool and its supporting infrastructure that we believe will fill the empty niche.

We found that most security tools provide either an internal view of a host or an external view of traffic on a network. Our interviewees revealed how they must construct a mental end-to-end view from separate tools that individually give an incomplete view, expending valuable time and mental effort. Because of limitations designed into TCP/IP [RFC-791, RFC-793], no tool can effectively correlate host and network data into an end-to-end view without kernel modifications. Currently, no other visualization exists to support end-to-end analysis. But HoNe's infrastructure overcomes TCP/IP's limitations bridging the network and transport layers in the network stack and making end-to-end correlation possible.

The capstone is the HoNe Visualizer that amplifies the users' cognitive power and reduces their mental workload by illustrating the correlated data graphically. Users said HoNe would be particularly good for discovering day-zero exploits. Our usability study revealed that users performed better on intrusion detection tasks using our visualization than with tools they were accustomed to using regardless of their experience level.

Defining the Problem

We believe that information visualization [Card, et al., 1999] technology can help system administrators wade through the tremendous amount of data they must review to ensure their systems are secure. 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. We 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. Next we surveyed the literature to find out if existing tools fulfilled the requirements expressed by system administrators. Finally, we conducted usability evaluations on our own tools to make sure that we were offering what system administrators truly needed.

Interviewing System Administrators

We 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 our results have given us insights that are probably accurate but are not easily quantifiable.

Four interviewees were our pilot group who helped us develop the interview protocol we used to interview the other twenty. Eleven of the interviewees also helped us with expert reviews of prototypes we had developed. We asked them for their biographical information, security-related duties, intrusion detection experience, tools they used in security monitoring, and what challenges they faced in keeping their organizations secure. A demographic summary of the interviewees appears as Table 1.

Interview Results

While the results of our prototype tests are presented in earlier papers [Ball, et al., 2004, Fink, et al., 2005], our interviews demonstrated that system administrators are not as homogeneous a group as we had initially thought. There were large differences in duties, tools, and techniques between those who primarily administrate servers and those whose main job is to assist users. There are also security officers, network analysts, and operations center specialists whose duties had much in common with other types of system administrators, but who had different areas of emphasis and different tool support needs. We would describe our "average" subject as a male with 14 years of experience (mean 13.39, stdev 5.25), charged with the care of ten or more servers with UNIX-like operating systems.

We learned that the single most important indicator of intrusions was aberrant communication patterns seen on the network or in host log files. The second largest indicator of problems was the appearance of suspicious processes on a host machine. We 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.

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

1. Process Analysis: Looking for unusual names or numbers of processes on a suspect host
2. File System Analysis: Looking for unusual or modified files, especially executables, dynamically linked libraries, kernel modules, and service configuration files
3. Activity (log) Analysis: Using the log files to piece together what may have happened to a host in the recent past
4. Vulnerability Analysis: Using common or recent exploits and vulnerability reports as heuristics to find intrusions

Considering the times and types of analysis together, we can see when a particular analysis method is most and least applicable. Within each analysis time, we ranked the relative importance of each type of analysis to our interview subjects (Table 2). We decided to concentrate on Activity and Process analysis because they are the most applicable overall.

We were surprised that most of the interviewees preferred text-based tools to visualizations and other high-end tools. This preference may reflect a population tendency toward low-level data analysis, but subsequent studies (especially [Fink, et al., 2005]) have demonstrated that our user community actually prefers visual solutions but has not yet found satisfactory ones. They typically did not trust tools that give an overview without showing the supporting details.

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

Legend: Most Important | Moderately Important | Least Important

Table 2: Relationship of times and types of diagnosis.

The administrators described how they identify suspicious communication patterns on the network (Activity analysis) and then manually trace those patterns back to a set of suspect hosts. On these hosts they would look at the processes running for suspicious activity (Process analysis). The users were determining whether communications were malicious based on what processes were initiating them. We realized that our users could save considerable time and mental effort if they could automatically trace packets seen on the network to processes running on their monitored hosts. While there are many tools in the literature and practice that claim to correlate host and network events, we were quite surprised to find that none actually did this packet-process correlation that was so fundamental to the work our users reported doing.

Key Conclusion: Packet-process correlation directly supports work typically done by our users, but no tools exist to automate this activity.

A Taxonomy of Available Tools

We reviewed the literature and current practice and organized the known tools by the context of the data they use and the way they present it. Figure 1 illustrates the resulting taxonomy that we use to compare tools. The fraction in the lower right corner of

Area (# Sub)	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 SAU/SAS, 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 infrastructure, usage, and planning.

Table 1: Demographic summary.

each category in the figure shows the number of interviewed system administrators who mentioned using a tool from that category.

The abscissa (x-axis) of the taxonomy diagram has four views of communication context that security awareness tools typically employ:

1. The internal host view (IH) that presents data internal to monitored host(s) without regard to network connections.
2. The networked host view (NH) that displays data that concerns only the monitored machines, but includes the broader context of their network connections.
3. The network view (NW) that presents traffic data in the context of a network or internetwork.
4. The 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.

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. The ordinate (y-axis) of the taxonomy diagram contains four ways security awareness tools may display information:

1. Text-based displays (TB) may have graphical user interfaces, but the information presented is pure text.
2. Dashboard-style displays (DA) present mostly text data, but use simple preattentive features like color and motion (blinking) to draw the user's attention to critical items.
3. Summary charts and graphs (CG) present abstract quantities like throughput pictorially via statistical graphs, etc.
4. Visualizations (VZ) convey the state of some object (a machine, a service, or an alert, for example) via an abstract marker or icon.

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. Our tool (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.

The Need for Visualization Across the Host/Network Boundary

Our discussions with system administrators regarding security incidents and common problems gave rise to several usage scenarios where visual packet-process correlation 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 visual packet-process correlation, we could see whether the clients and servers are really web browsers and web servers.

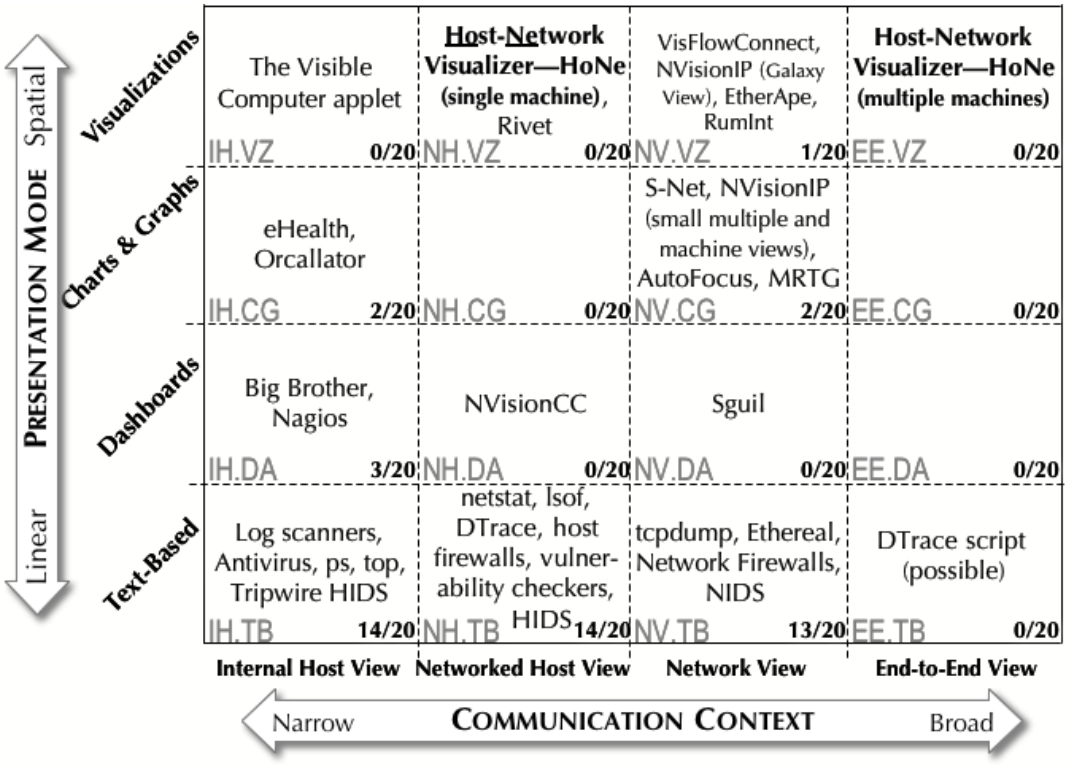

Figure 1: Taxonomy of security awareness tools.

- *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. It would be helpful to be able to observe the effects of traffic on the target host via visual packet-process correlation.
- *Cluster Computing*: Administrators who maintain large cluster computers could use visual packet-process correlation to see 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.

But Where are the Tools?

We surveyed tools of all kinds, free and commercial, and found our approach was unique. Here, we have only included a brief discussion of a few tools that are closest in apparent function.

Programs like Zone Lab's firewall, ZoneAlarm [ZoneAlarm], 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 [Foundstone] 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.

The lsof utility (http://freshmeat.net/projects/lsof/), 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, view of communications on the local host. 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 state of TCP and UDP sockets. On some UNIX-like operating systems and Windows, netstat can also tell what process the socket belongs to. Adding the process makes netstat comparable to Sysinternal's TCPview Pro [Sysinternals2]. Both lsof and netstat show communications from the host's point of view and both use polling. Thus, they may miss very short connections or communications that do not use sockets such as ICMP. Sysinternal's ProcessExplorer [Sysinternals1] is a Windows equivalent of a netstat/lsof amalgam, but does no more than either of these can.

The eHealth suite of network management tools from Computer Associates [eHealth] 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 networked host views or end-to-end views.

A staggering amount of effort is required to monitor computer activities for security. Most administrators we have interviewed care only about a few hundred machines they are personally 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. 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 correlates each packet to a process on the machine that sent or received it and provide a visual presentation of this correlation. Thus, our approach is unique.

Solution

There are many approaches to fill in the gap left by existing tools. In this section we present the alternatives we considered and the final solution we arrived at to correlate packets to processes and visualize the result.

Alternative Approaches

On investigating why no packet-process correlating software existed in the literature or practice we discovered that the separation between host and network was enshrined in the networking stack inside the kernel. The problem is illustrated in Figure 2. Given a modern operating system kernel and network stack based on a layered networking model [OSIArch, RFC-1122] (Linux, Solaris, *BSD, Windows, and MacOSX to name a few), we cannot know both the source machine and the associated processes for a given incoming packet at the same time and place.

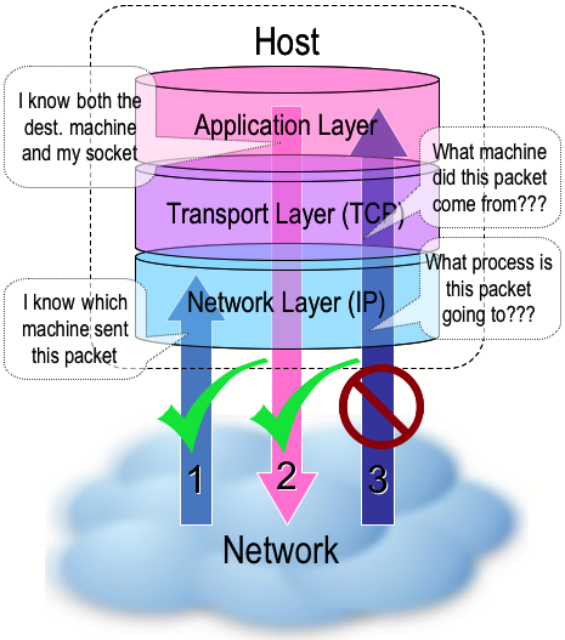

Figure 2: In network stacks based on the ISO layered networking model, we cannot simultaneously know both the source host and destination process for incoming packets.

This is not an oversight in the kernel design so much as it is an intentional separation of concerns characteristic of the layered approach [RFC-1122]. 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 necessary correlation could not be obtained in a faithful rendering of the standard protocol stacks.

For incoming packets, (number (1) in Figure 2) we know the source host at the IP layer only. When the packet reaches the transport layer (as in arrow (3) in Figure 2), this information is no longer available. In the application layer there is no problem for outgoing packets (number (2) in Figure 2) 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 we were loath to do. The following are the implementation alternatives we tried:

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

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 our 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 will 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.

To see whether or not netstat could capture all the information needed, we wrote a script that retrieved random web pages at normally distributed random intervals ($\mu = 5$ sec, $\sigma = 1.5$ sec). We collected and examined the netstat data at the maximum frequency while the script was running and found that whole connections were missed. We then modified the netstat source code removing the sleep interval to see if its coverage improved. The resulting program dominated the processor and generated 22 MB of text per minute when we ran it against only 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 we 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 appeared 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.

Modifying Socket Libraries

Another approach would be to modify the program libraries responsible for creating and maintaining sockets (particularly GlibC) to 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 an earlier usability study we found that bypassing these libraries was a common tactic used by malware authors to avoid detection.

Firewall Redirection

To ensure that we accounted for all the packets while the sockets were still active, we needed to move closer to the kernel. By adding packet redirection rules into the monitored host's firewall, we could copy all traffic to a special ''divert socket'' owned by an accounting process and correlate each packet to a running process or determine that no such process exists.

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. Reinserting the packet at the proper place in the firewall's rule set without changing the effective behavior of the firewall is very tricky. This method also contacts the accounting process for every packet that enters a monitored interface regardless of whether the machine is 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.

Kernel Modifications

Because none of the other methods worked, we were 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. We chose an implementation that had the smallest performance impact on the kernel. We 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. We then process the text file into an SQL database and visualize the data via a Tcl/Tk user interface. The performance impact to the machine is comparable to running tcpdump, and there is no impact when the kernel module is not loaded.

An alternative to the modified kernel that we did not try 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. A script using Sun's DTrace [Cantrill, 2004] may be capable of such kernel instrumentation without modifying the kernel at all. At the time we created the bridging architecture, we were unaware of DTrace, but using it instead of custom kernel modifications could be an excellent direction for future research.

HoNe's Architecture

We used Linux as our development platform because it is open-source and its kernel is relatively well documented. We planned for HoNe to be able to remotely visualize data from other machines by separating the user interface, data collection engines, and database functions into distributable components. Figure 3 shows a block diagram of the overall architecture.

The HoNe Visualizer is the key control component of HoNe. The user can use it to load and unload the loadable kernel module and to run the database builder to update the connection database from the connection log file or external data sources.

When the kernel module is loaded, it registers handlers for Netfilter hooks for all incoming and outgoing packets. Each incoming packet triggers the

Local_In handler. If the packet is a TCP or UDP (IPv4 only) packet the handler copies the IP header plus the first 20 bytes of the TCP or UDP packet to a log buffer, records a high granularity timestamp (via gettimeofday()), retrieves the process identifier (PID) of the socket owner, and queues the message for logging to the connection log file. The module performs similar work when an outgoing packet triggers the Local_Out handler.

Part of the difficulty with this approach is that Netfilter exists in the IP layer of the communication stack, while the information about the process that owns this socket is in the TCP layer. Thus, for incoming packets, we had to expose a private TCP function in the kernel dv_ip_v4_lookup() which retrieves a pointer to the TCP socket data structure. This pointer is readily available for outgoing packets. We added to the socket data structure a reference to the socket's owning process (actually its creator) that we fill in when the socket is created. We do the same for UDP, but we do not handle other protocols.

Whenever the kernel module is called, it checks to see whether there are any log records to write. If there are, it formats these and writes them to the log file. As part of the formatting process, the kernel module converts the socket owner's process identifier (PID) to the filename of the process's executable. This makes it possible to the user to locate executable files that are causing problems. The log records contain a timestamp, the PID and name of the owner process, the socket state, the packet size, and the packet header. We convert the packet header to a hexadecimal text string so the log file can be a text file. We could also log to an offsite analysis server via a socket, but we have not yet implemented this. When the kernel module is unloaded, it records the number of packets it processed and how many it dropped.

Description of a Successful Hack

In this section, we introduce HoNe by showing what happened on a monitored machine as it was hacked. This was a real incident, not a simulation. Figure 4 shows the state of the machine during the critical first 20 minutes of the intrusion.

In Figure 4, 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 we stopped the experiment. Because we can see

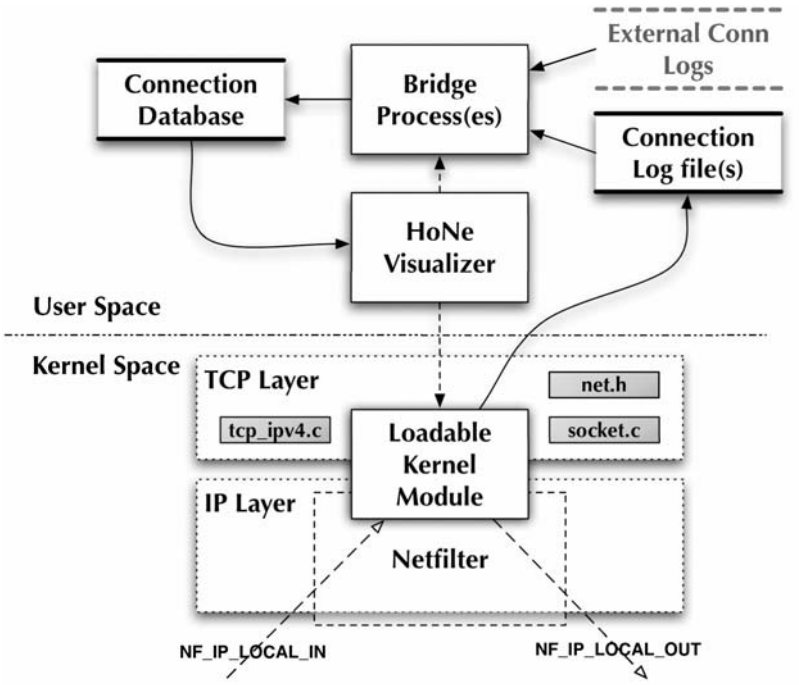

Figure 3: A block diagram of the HoNe architecture. The shaded blocks in kernel space indicate changes to the kernel itself beyond the loadable kernel module. Solid arrows indicate data flows and dashed arrows indicate control flows.

how the communication flows connect to processes, we get a clear picture of what is happening as a machine is being hacked. We see that the machine is talking to a remote server on port 8000 and that a non-standard "sshd:" process is running. But more than this, we see how these facts relate: that the fake sshd process is the one responsible for this communication. Thus, we can classify the communication as malicious with a greater degree of certainty.

The HoNe Visualizer

This section presents the HoNe Visualizer's user interface and explains its operation. The main window layout shown in Figure 5 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 Managed, and 4) Foreign Unmanaged. Machine icons appear in the upper (Enterprise) regions when they belong to the user's company. In the lower regions are the Foreign hosts. Machines that are administered and monitored by the user are considered Managed and appear on the left side. A machine should be considered managed if it is running the kernel module and is providing reports to the user. Machines on the right 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.

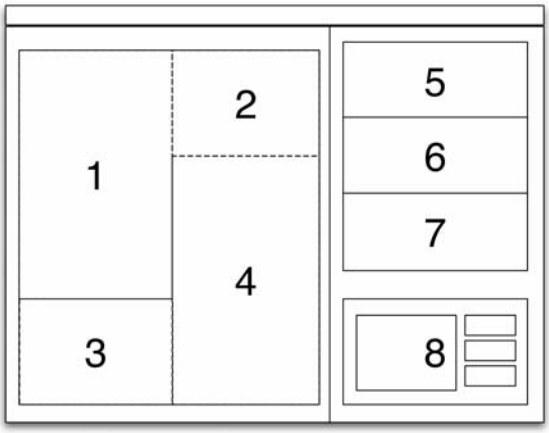

Figure 5: General layout of the HoNe main window.

The right-hand pane shown in Figure 5 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,

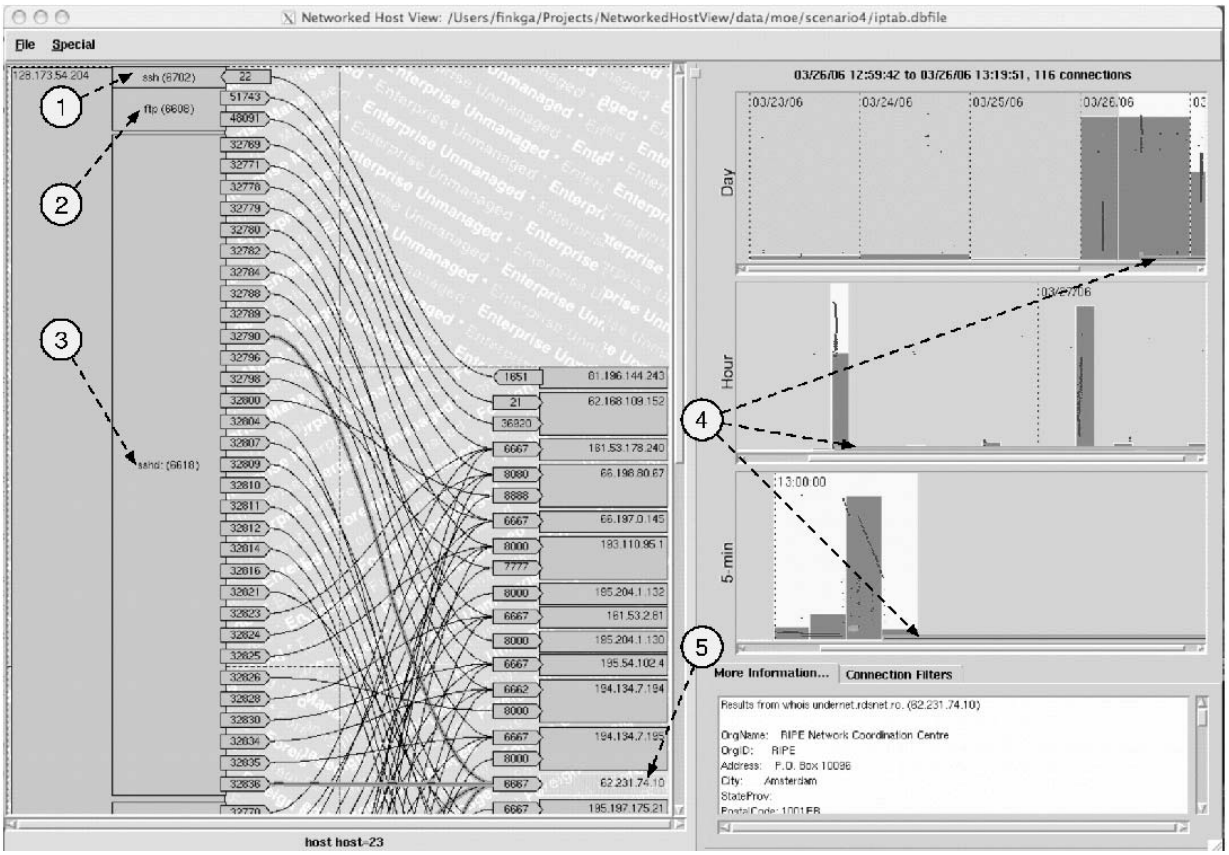

Figure 4: Snapshot of a successful hack: (1) Login, (2) downloading a toolkit, (3) starting the IRC 'bot, (4) suspicious, 23-hour connection, (5) Romanian IRC server contacted.

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, we will refer to another intrusion incident we captured using HoNe. Figure 6 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. We have elected to keep the machine icons together rather than to keep the time sequence intact.

First we see a barrage of SSH login attempts (highlighted points on the time window panes on the right) from foreign host 200.32.73.15. This barrage 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, we 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 three minutes, including downloading and starting the IRC bot, we believe this attack was automated. Full forensic analysis was not performed for this incident, so this is not a definitive analysis.

The detailed display (enlarged in Figure 7) shows each host by IP address (and DNS name if known). In Figure 7, 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 7). 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

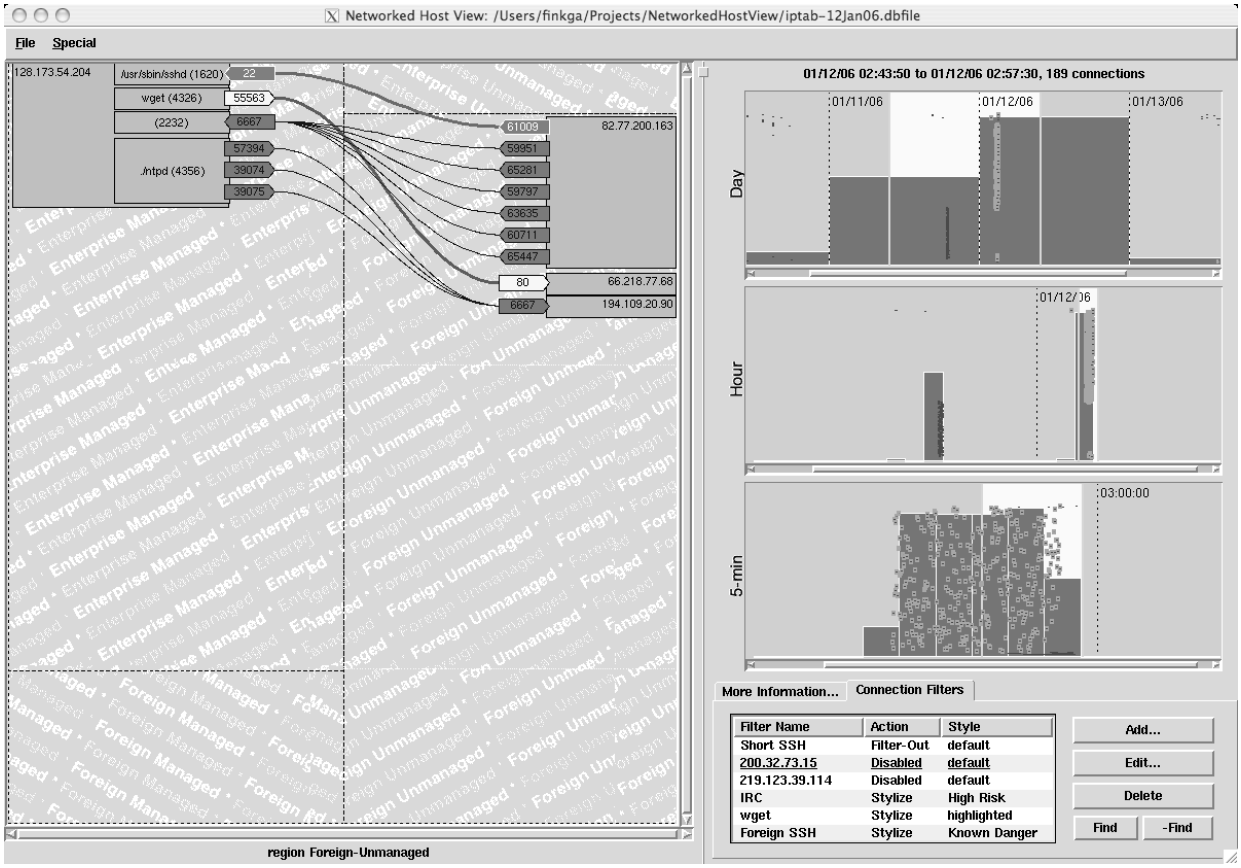

Figure 6: A screenshot of HoNe's visualization. The visualization displays connection data from an SQL database generated by the modified Linux kernel to visually correlate each packet to a running process.

Figure 7). Ports are shaped like arrows with listening (server) ports pointing toward their machine and initiating (client) ports pointing away. Item (3) in Figure 7 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 7) join the client and server ports. Item (4) in Figure 7 points to a connection between an ephemeral port on the client side and the managed host's SSH server port on the server side of the connection. The arrow-shaped port icons point away from the client and toward the server. Communication lines and icons for ports, processes, and hosts may be colored manually or by some user-defined filtering expression. 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).

HoNe provides multiple levels of drill-down so the user can keep the detailed view uncluttered. However, the nature of Internet activity makes it possible for thousands of activities to happen within a fraction of a second. Thus we enable users to zoom or shrink the icons in each region independently to fit them into a single screen.

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 8). The bottom section is a tabbed area where the Connection Filters and ''More Information'' panes reside.

The connection overview (Figure 8) 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 how we chose to span this large magnitude range.

The overview was inspired by histogram sliders [Li and North, 2003], 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. We 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 8 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.

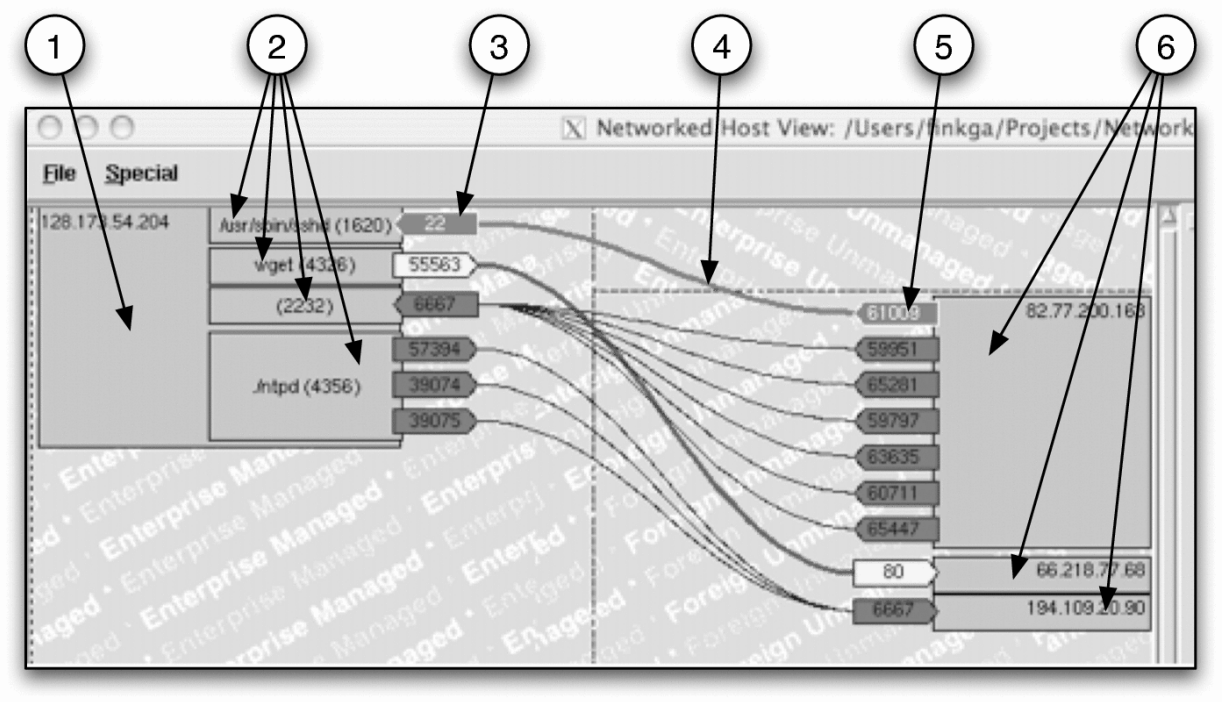

Figure 7: An enlargement of the detailed view of the HoNe visualizer.

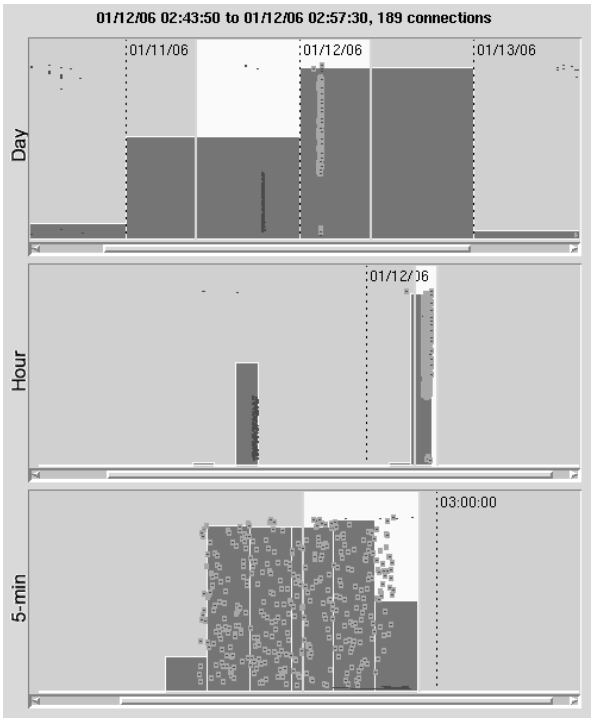

Figure 8: Detail of the connection overview histograms. Connection lines are brown but highlighted in cyan when selected. Selecting any connection highlights it in all views.

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 six 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.

The histograms are overlaid with horizontal dark brown lines that represent individual connections. The horizontal length of the connection line represents the duration of the connection. Because the scale may force many connection lines to zero length, we have constrained all lines to at least two pixels length. The longer duration a connection has, the nearer it is placed to the bottom of the histogram. The metaphor we use 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 8, 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 9) and Connection Filters (Figure 10) 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 9.

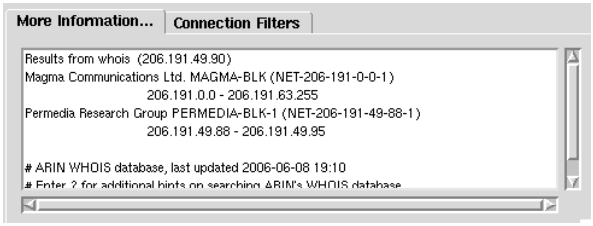

Figure 9: The More Information tab.

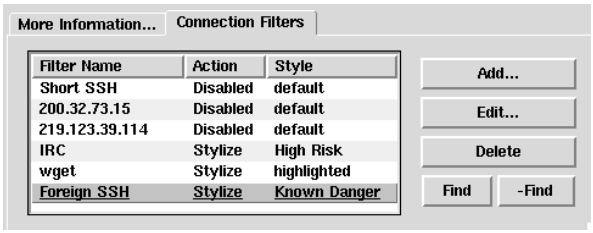

Figure 10: The Connection Filters tab.

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. Our users expressly requested each of these "more information" features during the pilot study.

The second tab of this section contains connection filters (Figure 10). Filters are SQL expressions the

user can employ to remove extraneous display items from the detailed view showing only 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, we 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 that area.

Users may define arbitrarily complex filters, but the best practice seems to be 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.

The user may define filter to 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. 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. We found the combination of powerful data filters and user-definable visual styles very useful for analysis.

Evaluation

Our 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. After building HoNe, we rigorously evaluated it to find out how well our objectives were met and what remained to be done. We designed the evaluation to answer the following questions:

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

We were 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.

Pilot Interviews

We conducted a pilot study with six expert computer security professionals and two information visualization experts. We selected subjects based on their known expertise and their helpfulness in prior interviews. We asked the pilot study subjects to honestly evaluate the visualization while using it to discover a real hacking incident in data collected using our kernel modifications. Their valuable insight helped us determine what pieces were still missing from the visualization and how much intervention would be necessary for less experienced users in the study to be able to use HoNe productively.

From the pilot study, we learned that HoNe had great potential but needed a little refinement. There were 26 enhancement requests, six areas where intervention would be needed, eight negative comments, and 11 positive comments. We used this feedback to refine the user interactions and to add options to provide more information about hosts, processes, and ports as the experts requested.

We found that the area needing the most intervention was constructing SQL queries for filters. Our interviews made it clear that our initial implementation of time windows based on slider widgets was unusable, so we replaced it with a more graphical approach that employs histograms, connection lines, and direct manipulation. We added the ''Find'' and ''Inverse Find'' buttons to help users rapidly locate

information of interest within huge files. The pilot interviewees told us that HoNe would be an effective tool for intrusion detection with powerful filtering and graphical capabilities that would be most effective for discovering zero-day (previously unknown) exploits. They also told us that our correlated data was a new and unique source of information that could be used in automated intrusion prevention systems.

Summative Usability Experiment

We designed a two-factor experiment, with the primary factors being *visualization* – at two levels (on or off), and correlation – at two levels (on or off). The four experimental conditions were:

1. No-Visualization/No-Correlation (NVNC): User has no visualization and only the uncorrelated text output of tcpdump, netstat, and lsof to work from (Control condition).
2. No-Visualization/Correlation (NVC): User has a human-readable text version of the correlated data from the kernel modifications 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, but the correlated data is not.
4. Visualization/Correlation (VC): User has all the data from the previous conditions along with a visualization of the correlated data.

We collected data for four scenarios using our modified Linux kernel. For each scenario, we simultaneously collected data from netstat, lsof, tcpdump, syslog, and our 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. The scenarios were each between 12 and 120 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.
2. Engineered Hack: We hacked the machine by logging into an unprivileged account under cover of an SSH scan, downloading a rootkit from a remote server, and starting 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 but real, not engineered by us.

There were 16 possible (condition, scenario) treatment combinations. We had each subject perform four runs under different treatment combinations such that each subject experienced every condition and scenario exactly once. We perturbed the order of treatment combinations to counter learning effects. Statisticians from an independent statistical consulting group verified that the experiment was balanced.

We recruited 27 subjects, about half of whom had more than a year of professional experience as a system administrator. We provided help and training on all needed skills (e.g., writing SQL queries or tcpdump packet filters), and told each subject the limitations of how much we would help them. The intervention and training brought all the subjects up to a certain minimum skill level to allow them to complete the scenarios. We were interested not in how well people used basic tools such as tcpdump and grep, but in how well various viewing conditions helped them see what was happening in a dataset.

In each run, we asked the subjects to identify intrusions or any other security-related features of the data. We established our "ground truth" about scenario events using a priori knowledge and the judgment of security experts who were given all the captured data (including data not available to the experimental subjects) at their disposal. We assigned each security-relevant feature a unique identifier and a value using a seven-point scale (Table 3). Subjects gained points for noticing and correctly diagnosing features but lost points for incorrectly diagnosing a noticed feature. Subjects had approximately 15 minutes to diagnose features for each of their four runs.

Points	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

Table 3: Seven-point Insight Score scale.

For each scenario and viewing condition, we totaled 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.

We 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. We used the normalized scores to compare scores across different scenarios. Then we subtracted the normalized negative score from the normalized positive score to obtain the insight score (Figure 11). We used the insight score to award prizes among other things. This approach prevents a user from attempting to increase his score by reporting lots of inconsequential features and reduces the likelihood of guessing. Accuracy is the most important characteristic of the kind of work we are 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.

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

Figure 11: Formula for Insight Score.

Data Collection

We wanted interviewees to see real data from actual break-ins, so we set up a sacrificial host on the campus LAN and outfitted it with Snort and Tripwire (two freely available intrusion detection systems). Then we created a User Mode Linux (UML) [UML] virtual machine equipped with the modified kernel running in a process on the real machine. We bridged the real and virtual machines' Ethernet interfaces so that they looked like separate machines on the same LAN segment.

A defect in our kernel modifications caused the machine to crash when attackers attempted to subvert the machine in certain ways. Kernel-level rootkits such as Adore [Adore] and LRK5 [LRK5] caused the machine to crash when they were activated. Thus, after the attacker initially penetrated the machine, his attempts to install a rootkit would crash often 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, the defect often prevented us from collecting data beyond the initial breakin.

Summative Study Findings

Users preferred the correlated visualization (VC) and felt they got the most insight from it of any of the conditions. The VC treatment resulted in better insight 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, we showed that VC was better than NVC with marginal statistical significance. From these findings, we infer that our visual correlation of packets to processes does help administrators perform certain intrusion detection tasks better than text data alone could.

The VC condition garnered a large number of unsolicited positive comments about the visualization tool. We believe that users would have performed even better and had a more satisfying experience if not for some implementation bugs in the visualization and its suboptimal responsiveness.

The NVC condition provided the critical element of packet-process correlation without providing a means of visualizing the data. The correlated data was compact and contained in 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 made reading packet traces easier for our users. The correlated data would have been much more useful to humans if we had put a start time and duration instead of start and end times that the users had to mentally subtract to get duration. NVC users mentioned many times that they were looking for long sessions in the text. Those who had performed a run with the visualization previously often said that they most missed the visual indication of duration.

The VNC condition turned out to be very troublesome both to users and to us. 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 preferred and resulted in the lowest scores on average. Most users found this condition confusing and error-fraught. Novices performed very poorly with the staggering amount of data this condition presented. Many times novices would painstakingly try to understand each packet or connection attempt often covering no more than the first 1% of the data during the whole 15-minute run. We encouraged 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 in practice.

Conclusions

We have noted how the literature is divided between display of host data versus display of network data. We have shown that this disconnect is inherent to the TCP/IP networking model [RFC-791, RFC-793] and is codified into the kernels of all modern operating systems. Thus, automatic process-packet correlation has been impossible and was overlooked as an approach to security awareness in both literature and practice. But we have presented a solution that bridges the gap between the network and transport layers, enabling correlation of incoming packets with the processes that accept them. We have created the HoNe Visualizer as a user interface to this new correlated data and demonstrated that both experts and novices perform better on intrusion detection tasks with it than with text-based tools alone. Our research has advanced the science of computer security in several significant ways. To recapitulate what we believe are the most important contributions of our work:

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

Future Plans

In the future we plan to replace the kernel modifications with a set of DTrace scripts. This will make gathering the correlated data safer and more portable. We plan on improving our visualization and making it more efficient, but other visualizations or back-end process could use this rich new source of data to increase the security of monitored machines and inform rapid and accurate responses to communications problems.

We believe our host/network bridge and visualization will be a valuable asset to system administrators. This type of visualization will amplify the insights and abilities of system administrators, complement the existing tools they use, reduce monitoring costs, and increase the security posture of organizations that use it.

Impact

HoNe has demonstrated how helpful packet-process correlation and visualization are for detecting and diagnosing potentially malicious activity on computers. We 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 [Cantrill, 2006] 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 we have done with HoNe: we modified the kernel to gain visibility into how packets relate to processes and created a visualization of the information we gathered. Since lack of observability is a huge problem in today's system software, we expect to see efforts such as DTrace [Cantrill, 2004] gain broader acceptance and become available on more platforms. But these programs only dump more text at the user. What we 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, but it is our hope that HoNe will lay the groundwork for a positive change in the way security professionals go about their work. If kernel designers accept our conclusions and incorporate greater observability of the packet-process relationship into their work, much better security monitoring will be possible in the future than 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.

Finally, we hope that the success of our work can demonstrate how important it is to talk to system administrators and involve them as co- designers in any work that purports to meet a need. Tools must adapt to their user's needs to be truly useful. The important thing is to find out what the users need (even if it is not what they actually asked for) and then design tools to fit the need. HoNe is one such tool-may many others follow it.

Acknowledgments

The authors would like to thank Virginia Tech's security officer and the system administrators we interviewed for their patience, enthusiasm, and guidance as we shaped this product to fit their needs. This research

was supported in part by a National Science Foundation Integrated Graduate Education and Research Training (IGERT) grant (award DGE-9987586).

Author Biographies

Dr. Chris North, Associate Professor of Computer Science at Virginia Polytechnic Institute and State University, is head of the Laboratory for Information Visualization and Evaluation and member of the Center for Human-Computer Interaction. He received his Ph.D. at the University of Maryland, College Park. He co-led the establishment of Virginia Tech as an NSA National Center of Academic Excellence in Information Assurance Education. His current research interests are information visualization, high-resolution displays, and visualization evaluation methods. Contact at north@vt.edu, http://infovis.cs.vt.edu/ .

Glenn Fink recently completed his Ph.D. at Virginia Tech where his dissertation was on visual correlation of network traffic and host processes for computer security. At this writing he is moving out to Washington state to accept a job at Pacific Northwest National Labs where he will work on more visualization technologies for computer security applications. Reach him via email at finkga@vt.edu .

Ricardo Correa received his BS in Computer Science from the University of Texas at El Paso. He worked for the University's IT department managing the campus-wide network infrastructure. He is currently pursuing an MS in Network Engineering at the University of Pennsylvania. Reach him electronically at ricm@seas.upenn.edu .

Vedavyas Duggirala is a currently pursuing his Ph.D at Virginia Tech. His work is in the area of large scale network emulation. He can be reached via vduggira@vt.edu .

References

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

[Cantrill, 2004] Cantrill, B., M. Shapiro, and A. Leventhal, "Dynamic Instrumentation of Production Systems," *Proceedings of the 2004 Usenix Annual Technical Conference*, 2004.

[Cantrill, 2006] Cantrill, B., "Hidden in Plain Sight," *Queue*, Vol. 4, Num. 1, pp. 26-36, http://doi.acm.org/10.1145/1117389.1117401, Feb., 2006.

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

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

[Fink, et al., 2005] Fink, G. A., P. Muessig, and C. North, *Visual Simultaneous Correlation of Host Processes and Network Traffic*.

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

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

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

[OSIArch] Zimmermann, Hubert, "OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, Vol. 28, Num. 4, pp. 425-432, April, 1980.

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

[RFC-791] RFC 791, "Internet Protocol," September, 1981 (See also: MIL-STD-1777).

[RFC-793] RFC 793, Transmission Control Protocol, September, 1981 (See also: MIL-STD-1778).

[Sysinternals1] ProcessExplorer, http://www.sysinternals.com/Utilities/ProcessExplorer.html , Last accessed September, 2006.

[Sysinternals2] TCPViewPro, http://www.sysinternals.com/Utilities/TcpView.html , Last accessed Sep., 2006.

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

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