HoneyTrap: Visualization for Monitoring Honeynets

John E. Clark, Christopher P. Lee, Roshan Menon, Vanessa Rood
Georgia Institute of Technology

ABSTRACT
With the sharp rise in computer network attacks, security personnel must quickly react to impending threats in order to mitigate damage. In this paper we introduce HoneyTrap, a passive visualization for monitoring Honeynet alerts. This passive visualization provides notification of attacks without distracting the user with huge volumes of data that can result in alarm fatigue. The HoneyTrap provides a quick view of the informational goal, which is simply: Has a machine been attacked and/or compromised? In addition, details on demand allow the system administrator access to a deeper level of information regarding individual attack specifics. In this paper, we focus on the monitoring tasks of a Honeynet administrator and which visualization techniques were choosen to incorporate into HoneyTrap to accomplish these tasks.

Keywords
Information visualization, intrusion detection, Honeynet, Honeypot, passive, security visualization, network attacks, monitoring

INTRODUCTION
Typical network traffic generates a huge amount of data for analysis. Determining if traffic is suspicious can take a significant amount of time and effort. Due to the number of networks and attacks represented with existing intrusion detection systems, it is often difficult to determine where legitimate problems are actually occurring. Several tools have been created to analyze and sift this data. These security tools often require a great deal of configuration in order to provide accurate alerts for suspicious activity.

In many cases, a honeynet can serve as the “canary in the coalmine” by providing a first alert of suspicious traffic. A honeynet is a small, connected network of honeypots. A honeypot is an unused computer placed on the network without protection. The honeypot acts as a decoy of sorts, to purposefully attract questionable and possibly dangerous network activity. Because the computer is unused, any network connections made to it or from it are immediately suspicious. It is purposefully unprotected to allow the attacker to connect to the machine and to compromise it.

When an attacker compromises the machine, all traffic is recorded to capture how the machine was compromised, how the machine “looks” when compromised, and what the attacker does after the machine is compromised. All connections are monitored to determine what type of internet attack was attempted. The most common attacks are: web exploits, worms, ssh brute force, and botnets. Depending on the nature of the attack, different measures are needed. If a honeypot is compromised and is being used as part of a botnet, it must be disconnected quickly. If an attacker has been able to log into a computer, but has yet to do anything interesting, then extra monitoring is needed.

To this end, we have developed HoneyTrap, Figure 1, to visualize the state of the honeynet; specifically whether an interesting attack has occurred. HoneyTrap enables these tasks using a passive visualization technique to quickly show “at a glance” the state of the entire honeynet.

In the first section of this paper, we will examine a number of related visualizations to see their strengths and weaknesses. Then in the Tasks sections we will expand on the visualization tasks required for monitoring the honeynet and how the HoneyTrap addresses these tasks interface. In the final Interface section we will describe the HoneyTrap interface in greater detail.

Related Security Visualizations
VISUAL [3] uses only packet traces for its data source, specifically preprocessed PCAP files. The tool can be used for forensic analysis of packet data for a subnet consisting of less than 1000 hosts. Port scans and ping scans are easily recognizable as long as there is not a lot of other irrelevant traffic. This system is good for delineating general communication patterns. It is not suited for recognizing malicious activity, because it does not use any system logs or IDS alarms.

Figure 1. Screenshot of the HoneyTrap passive visualization display with multiple alarms
Rumint [4] is used for real-time monitoring of network traffic. It uses parallel coordinate plots to show traffic patterns between various hosts on a network. This tool is designed to passively fingerprint network attack tools. Instead of attempting to fingerprint network attack tools, our Visual Signature view fingerprints the behavior of a host (traffic pattern) during and after a security incident, such as infection by a worm or a distributed denial of service (DDoS) attack.

SecVis [10] is a visualization tool for real-time and forensic network data analysis. This tool displays packet capture data as a 3D parallel coordinate plot along with a dynamic scatter plot. Some network attacks are very apparent, but this tool does not take into account IDS data or system logs.

Figure 2. Snort View IDS alarm visualization

SnortView [9], seen in Figure 2, is a tool that was developed specifically for analyzing Snort logs and syslog data. Its primary purpose is to use visualization to more effectively recognize false positives. It presents an updated view every two minutes and shows four hours worth of alert data. One slight limitation is that its user interface is in Japanese.

PortVis [13] only analyzes high level summaries of packet data from a large network. Its primary focus is to detect large scale network security events. It provides multiple views of the same information to help correlate data and allow an operator to mentally shift between visualizations. The utility of this tool’s multiple views is one of our motivating factors for presenting multiple views in our visualization software.

Mielog [14], seen in Figure 3, was made specifically for forensic analysis of system logs. It uses statistical analysis for classifying log entries and visualization techniques for displaying different characteristics of the logs. The main goal of this tool is to manually parse logs, not necessarily visualize their content.

Erbacher’s Hummer IDS Visualization [7,8] uses a collection of logs and other network data from the Hummer IDS in order to represent network events between a monitored system and other hosts. Using real-time or forensic analysis, interacting hosts are visualized as a spoke and wheel diagram.

The Spinning Cube of Potential Doom [11] represents Bro IDS alarms (which include every completed and attempted TCP connection) as colored dots in a 3D spinning cube. In this perspective, the X and Z axes represent local and global IP addresses, while the Y axis depicts port numbers. Network attacks have obvious visual illustrations; for example, port scans are displayed as linear lines.

The Analysis Console for Intrusion Databases (ACID) [6] is devised for active analysis of Snort logs. ACID uses a web based interface to present alerts as charts and graphs in HTML. However, administrators must still peruse intrusion alerts in their native text format.

VisualFirewall [10] combined several real-time views of network traffic along with feedback from the firewall system to visualize attacks on a given system. The use of multiple views makes the visualization resilient to malviz [5]. However, the visualization does not scale to larger network sizes.

IDS Rainstorm [2], Figure 4 (next page), uses a rainfall visualization to show 24 hours worth of IDS alarms and uses panning, zooming, and filtering to allow the administrator greater detail of network activity. This allowed for a large amount of pre-attentive preprocessing and pattern detection within the large volume of alerts generated by the IDS. This visualization is not real-time, and with a time resolution of 20 minutes, updates are rare and barely noticeable. Additionally, no historical data beyond the 24-hour period could be displayed.
Figure 4. IDS Rainstorm visualization of 24 hours of alarms

Related Passive Visualizations

InfoCanvas [16] uses a passive visualization to represent information relevant to the user (similar to the information that might be found in a typical web based personal start page). The source data and visualization components can be defined by the user, which creates a representation that will be easily familiar to the user. Typical examples of data sources used in InfoCanvas include local weather, traffic, unread emails, airline prices, and stocks. These data items are then mapped by the user to various information markers on the screen. For example, in Figure 5, the lady’s suit color corresponds to Atlanta’s traffic (green here only for illusion of good traffic in Atlanta).

Figure 5. InfoCanvas passive display of data indicators

Stasko et al., performed several months of user studies to determine how people used the visualization and how effective it was in conveying useful information. They showed that people have a statistically significant higher recall rate of information compared to a web portal containing the same information and that people were generally pleased to have and to use this visualization.

The Informative Art project [22] seeks to create informational presentations that can be relegated to the background of the users perceptions when not in use, but can be read and understood when needed. Using an appearance similar to museum artwork, these visualizations modify themselves based on the information being gathered (bus routes, email content, meeting times, etc.), providing information to the user that is non-intrusive, but still informative.

Figure 6. Informative Art representation of bus line 16 in Gothenburg, Sweden.

VIZUALIZATION TASKS

In order to quickly react to threats, notifications of honeynet activity need to be immediately sent to the security personnel. Security administrators are most interested in how the machines were compromised and how to protect the remainder of their networks. The security administrator needs to know which services are being attacked. Because the vulnerable parts of the computer's security are the services it runs, it is important to map the observed network traffic to the most likely service under attack. While this is not always clear, current practices give security administrators the details needed to make security decisions for firewalling, repairing compromised machines, and future policy.

Additionally, security researchers are interested in how the exploit works and the motivations behind the attack. These motivations are deduced from the actions taken after the compromise.

The critical piece for the Honeynet administrator is seeing when an attack happens and getting a rough idea of the nature of both the attacker and the attack. The HoneyTrap allows security administrators to easily monitor the status of honeynet activity by facilitating the following tasks:

1. Pre-attentively discern new and/or abundant activity on the network.
2. Provide an ambient feel for the amount of activity on the honeynet.
3. Determine, with little attention, which classes of attacks are occurring.
4. Supply enough "history" in the visualization that security admin can temporarily leave their desk without worrying about missing a successful security breach.
5. Deliver an immediate alert for the present administrator when a successful attack occurs.
6. Drill down and get a detailed set of information on specific attacks – e.g. IP addresses of attacked honeypot and attacker, port activity, attack type, time stamps, country of origin.

The Use of Color
For this visualization, we use colors to show state, highlight areas of interest and to show groupings and patterns [20] (Task 3). For areas in which the use of color might appear ambiguous with a quick glance (such as the different color of bees for each country), the color relationship information can be made clear by clicking on the bee to see the details.

The attack colors are also displayed in the details section for easy reference. This legend serves a dual purpose as both a recall aid and a set of toggled buttons for showing and hiding certain attack types. This way, if the display should become too "noisy" the administrator could easily turn certain attacks off as needed, allowing him/her to filter attacks by type as needed.

By making the active honeypots a stark white color, they easily stand out from the unused background hexes. The color contrast also serves to group the hexes visually. Attacks are easily seen as colored hexes replacing the plain white hex of an active honeypot. This draws the user’s attention by changing both the content and color of the visualization which has been shown to be more easily spotted then the change of simply one or the other [20] (Task 5).

With the exception of clearing alarms and attack notification, we have avoided the use of animation, therefore, preventing the user from being unnecessarily distracted. [15].

Passive Display
Why passive? Typical visualizations in the security field tend to focus on the analysis of network traffic and aid in determining whether or not the traffic is hostile. In the case of the Honeynet, all traffic is suspicious. This fact lends itself to a binary type of visualization (i.e. is someone trying to get into the Honeypot or not?) and allows us to display a small subsection of data that would be useful to a honeynet administrator.

In this case, the value of the visualization is in the notification of the attack and the easy access to a higher level of detail [19]. The details will not paint the whole picture for the Honeynet administrator (that is beyond the scope of a passive tool), but they provide enough information to let the administrator know that something has occurred and where to begin the search so they can deal with the attack. A combination of the passive display system with an active details section provides a number of benefits in this regard.

First, with the passive display, the data is available at all times without causing undue distraction [16]. The display can be pushed to the mental background when not needed. However, the appearance of a new item on the display is enough to attract the attention of the administrator.

Secondly, the graphical interface of the HoneyTrap lends itself to a passive display. The metaphor of bees and honeycomb hexes relates directly to the terminology of ‘honeynets’ and ‘honeypots’. Subsequently, these graphics provide an easy mental model for the user and a consistent display of information in a simple and easy to recall format. The variations in color allow for easy comparison of data without distortion. They also reveal a greater detail then the simple binary state of ‘attack’/’no attack’ as well as providing an aesthetically pleasing graphical display [18].

Passive displays have also been shown to provide a high rate of data recall in test subjects [16]. This recall rate will aid the Honeynet administrator in consistently reading the data as presented in the HoneyTrap project. We have also used a very small set of visual tracking information (attack, type and country), which gives the system administrator only a few pieces of initial data to monitor.

We also allow the layout of the hexes to be controlled by the Honeynet administrator. The screen layout can be customized by individual administrators, allowing them to group the honeypots on the screen according to their preference. This ability further aids in the recall of the data being represented.

Combining the passive visualization with the addition of an active details section allows the administrator to quickly comprehend additional information and to help with understanding the nature of the attack. This details section also reinforces the information contained within the passive visualization, providing details on demand by clicking on an attack (Task 1 & 6).

The timeline view at the bottom of the interface allows the administrator to leave the visualization unattended while still maintaining a history that can be referenced later (Task 4). Visual alerts accumulate along the bottom of the 24-hour timeline, corresponding to the time the alert was generated. Additionally, if the administrator has cleared the alert, the time that it was cleared appears along the bottom
of the timeline. The slider can be used to scroll back through the timeline and see the status of the visualization at any desired time within 24 hours.

**THE HONEYTRAP INTERFACE**

For the HoneyTrap visualization, we represent the honeynet as a space of hexagonal cells, each cell representing a computer (honeypot), grouped together by organizational, physical locality, or any customized layout of the system administrator’s choosing. Each hexagon can be assigned to a honeypot by clicking on the hex cell and entering in the IP address, allowing Honeynet administrators to customize their layout however they like.

Colors are used to encode the state of the node, i.e., the attack that succeeded against the honeypot. We use icons of graphic bees to represent attackers. The bee encodes the attack origin and whether or not the attack has been acknowledged and/or cleared. When either the bee or a cell is clicked on, details about the state of the honeypot and about the attacker are given.

For the passive visualization to work it must be visually engaging and informative. It must also provide the relevant data in a clear manner with easily understood mappings. Additionally, HoneyTrap provides an active layer in the display that allows for detailed data and the ability to clear information that is no longer relevant.

The HoneyTrap visualization uses representative honeycomb and bee graphics as indicators for attacks. These graphic representations provide a clear mental model that fits nicely within the framework of “honeynets”, “honeypots”, and “attacks”. Additionally, the interface affords a whimsical quality that fits within the cultural framework of our user space.

When an attack occurs, the corresponding hexagon changes color to denote one of six common attacks (SSH, Botnet, MySQL, Worms, Web, MSSQL). A bee will also appear on the hexagon denoting one of sixteen countries or a plain yellow bee denoting "Other" for any country not in the sixteen currently supported (it is likely that both attacks and countries would need to be added to and possibly made a configurable option in future versions). A legend at the bottom shows the color for each attack. Multiple attacks on a single honeypot will be represented by the appropriate number of bees and the hexagon color will turn gray.

In order to provide more detail, a honeypot which is under attack may be clicked on and a detailed view window will appear containing information including IP addresses, country of the attacker, ports, the both the start and end time of the attack.

From this ‘Details’ window, attacks can also be cleared allowing the Honeynet administrator to respond to an attack and clear the graphic from the display after the relevant data has been gathered.

In some cases, a Honeynet administrator may need to be away from the display for some amount of time. As attacks occur, they are also added to the timeline at the bottom of the visualization. The timeline allows the Honeynet administrator to get a quick at-a-glance view of how long ago these attacks occurred. Again, more detailed information can be seen in the details window.

**CONCLUSION**

We have identified tasks in monitoring and responding to threats on a honeynet, and applied several visualization techniques to aid the Honeynet administrator. Through the combination of passive visualization, color coding, history sliding, customized mappings, and details on demand, our techniques avoid information overload and alarm fatigue. Furthermore, we developed a prototype, HoneyTrap, that is in current use by the Georgia Tech Honeynet administrator to monitor the local Honeynet project.

**FUTURE WORK**

We want to explore the possibilities of passive sonification of honeynet state to allow an administrator to monitor the honeynet without allocating a second computer display.

We would also like to explore visualization of incorporating long term trends into HoneyTrap so that the administrator obtains a sense of how current patterns are similar or differ from historic trends of activity on the honeynet.

**ACKNOWLEDGMENTS**

We would like to thank Dr. John Stasko and Bob Amar at Georgia Tech for instruction and feedback. Thanks also to the entire Spring 2006 Information Visualization class for their constructive suggestions.
REFERENCES


20. C. Healey, Perception in Visualization, Department of Computer Science, North Carolina State University Website: http://www.csc.ncsu.edu/faculty/healey/PP/index.html
