Countering Security Analyst and Network Administrator Overload Through Alert and Packet Visualization

Gregory Conti  
College of Computing  
Georgia Institute of Technology  
conti@cc.gatech.edu

Kulsoom Abdullah  
Department of Electrical and Computer Engineering  
Georgia Institute of Technology  
kulsoom@ece.gatech.edu

Julian Grizzard  
Department of Electrical and Computer Engineering  
Georgia Institute of Technology  
grizzard@ece.gatech.edu

John Stasko  
College of Computing  
Georgia Institute of Technology  
stasko@cc.gatech.edu

John A. Copeland  
Department of Electrical and Computer Engineering  
Georgia Institute of Technology  
john.copeland@ece.gatech.edu

Mustaque Ahamad  
College of Computing  
Georgia Institute of Technology  
mustaq@cc.gatech.edu

Henry Owen  
Department of Electrical and Computer Engineering  
Georgia Institute of Technology  
henry.owen@ece.gatech.edu

Chris Lee  
Department of Electrical and Computer Engineering  
Georgia Institute of Technology  
chris@ece.gatech.edu
ABSTRACT
When given the task of securing a network, security analysts and network administrators typically face large volumes of security data that demand analysis. Selectively mapping elements of these flows to carefully crafted graphical displays can provide rapid insights while, at the same time, actively counteracting information overload. To this end, we present a generic framework for designing such visualization systems as well as results from the end-to-end design and implementation of two highly interactive systems. The first system focuses on increasing the utility of intrusion detection systems by providing information rich displays of network alerts. The second system provides new methods of visualizing network packets that enable the analyst to efficiently and effectively explore network traffic for malicious activity. To support our findings, we present the results of a user requirements study, which we used to base our designs, and an evaluation of our completed systems by professional security analysts.


General Terms: Security

Additional Keywords: IDS alarms, alert visualization, payload visualization, packet visualization, log visualization, alarm visualization, network visualization, network monitoring

1 INTRODUCTION
The massive amount of security data generated by network sensors and host-based applications can quickly overwhelm the operators charged with defending the network. Often, important details are overlooked and it is difficult to gain a coherent picture of network health and security status by manually traversing textual logs, using command line analysis scripts or traditional graphing and charting techniques. In many instances, this flood of data will actually reduce the overall level of security by consuming the operator’s available time or misdirecting their efforts. In extreme circumstances, the operators will become desensitized and ignore security warnings altogether, effectively negating the value of their security systems.

We address this problem by carefully crafting graphical systems designed to present the data in insightful ways that tap into the high bandwidth visual recognition capability of human operators. We began our work by surveying professional security operators to determine the limits of today’s best systems and identify high payoff targets for improvement. Using these requirements to drive our designs we created two security visualization systems. The first system, IDS RainStorm, is designed to provide high-level overviews of intrusion detection alerts. The second system, RUMINT, is designed to provide detailed insights into packet level network traffic. We have deployed these systems in a variety of laboratory and operational settings for a total of two years to evaluate their effectiveness. During this period, we iteratively improved their designs and developed a general framework for designing such systems. In this article we provide multiple contributions: we present the design framework, results for our survey of security professionals, lessons learned from the design of our systems as well as an evaluation of their effectiveness. Our results indicate that both systems effectively present up to two orders of magnitude more information when compared to traditional textual approaches. We believe that the interactive, graphical techniques that we present will have broad application to other domains seeking to deal with problems of information overload. This article is based upon a series of conference and workshop papers which describe earlier versions of our work, most notably [1, 2]. We have built upon our previous work including a more developed framework, results from our user requirements survey, updated design, implementation and application details of our two systems as well as an expanded evaluation.

2 EVALUATING CURRENT BEST PRACTICES
Information overload is an everyday occurrence for security analysts. As an example, consider the day-to-day operation of the Georgia Institute of Technology’s campus network. At this institution, the total campus population is approximately 15,000 undergraduate and graduate students and approximately 5,000 staff and faculty. There are 69 individual departments spread over the campus with between 30,000-35,000 networked computers operational at any given time. The total number of Internet Protocol (IP) addresses allocated to Georgia Tech is equivalent to 2.5 Class B networks or 163,840 addresses. The network connection from the campus to the Internet has an average throughput of 600Mbps. On average, over four terabytes of data is processed each day by the network. The security, health and welfare of the campus network are the responsibility of the Office of Information Technology (OIT). Staffed by a handful of network analysts and security experts, OIT’s main concern is, by necessity, determining the location of only the highest priority security alerts and effectively allocating their limited human resources to resolve the problems. Unfortunately, the intrusion detection sensors deployed across campus generate an average of 50,000 alarms per day. In order to prioritize these alarms, OIT analysts typically resort to only high-level statistics, such as the number of alarms, alarm
severity and the time of day of each alarm. Intrusion detection tools come with limited visual components, and they have proven problematic to calibrate. As a result, browsing through textual alarm files and processing via command line scripts are usually the methods used to cope with the overwhelming volume of data. Currently, alert prioritization consumes the majority of the analyst’s time, leaving little time for analysis and response.

The problem is larger than just the Georgia Institute of Technology campus. As part of our research, we conducted a survey of professional security analysts working in industry as well as graduate students specializing in information security. There were 39 participants. Our survey studied the limits of two open source tools in very wide use: Snort, an intrusion detection system, and Ethereal, a protocol and packet analysis tool.

![Figure 1: Results from survey of expert security analysts from academia and industry. Figure depicts threshold at which human overload occurs when analyzing network packets (dark gray) and intrusion detection system alerts (light gray).](image)

Figure 1 shows that the majority of the survey participants start to become overwhelmed when the number of intrusion detection alerts reaches only tens of alerts per hour. Similarly, in packet analysis, overload occurred when faced with only hundreds to thousands of packets. Participants were explicitly asked to describe when they, as humans, became overloaded and not to respond based on the resource limitations of their given hardware platforms. It is useful to examine the thresholds by analyst background. Information security students had a lower capability to cope with alerts, reaching overload at an average of 30.67 alerts per hour (SD=35.70, N=6). Professional security analysts were able to handle more alerts than students, but most still faced overload at hundreds of alerts per hour (Average 229.33, SD=568.42, N=12). The situation is similar for packet analysis. Information security students felt that Ethereal became difficult to use when the number of packets exceeded hundreds of packets per hour (Average 494.44, SD=694.82, N=9). Professional security analysts were able to handle a larger number of packets without overload, but most respondents felt that Ethereal became difficult to use when faced with data sets on the order of thousands of packets (Average 6,631.25, SD=9809.36, N=16).

To clarify why overload occurred, study participants were asked to elaborate on the causes. Comments collected from the survey instrument were free form, but by clustering responses the following trends emerged:

**Snort (N=15)**

- 40% found analysis and ease of use difficult, particularly the graphical front ends available for the tool.
- 60% were concerned that intrusion detection signatures were easy to bypass and difficult to tune effectively for a given network. Of these, 56% felt that an attacker could easily bypass existing signatures and 44% found it difficult to effectively tune the intrusion detection system to minimize false alarms and missed attacks.

**Ethereal (N=27)**

- 59% indicated problems with the ability to see the big picture in the data due to overwhelming detail.
- 44% stated that the textual representation of the data was not up to the task and that the GUI became unmanageable when working with large data sets.
- 18% found the filtering capability difficult to learn and use.

The two systems we present in this article directly address the majority of these issues. It is important to note that we do not address the users’ stated problems associated with Snort’s signature matching and tuning. We believe this issue is more an artifact of the underlying signature based intrusion detection model. While resolving the shortcomings of signature matching intrusion detection systems is an important area for future research, we focus instead on addressing the problem of information overload. Therefore, our systems do address the remaining issues for both Ethereal and Snort and seek to improve existing best practices by providing insightful high-level overviews and detail on demand to support analysis of intrusion alarms and network packets. While we believe our systems are addressing real world needs when compared to today’s best-in-class operational systems, they also advance the state of the art when compared to the research literature.

Several tools have been developed to visualize and process Snort IDS alarm log files. One is SnortView [3] where a matrix view is used to show IP address connections over time. This tool is successful in combining multiple parameters, visually representing them to assist analysts in finding anomalous behavior, but the amount of information shown is limited to a subset of IP address ranges, time (4 hours) and number of attacks. Other related tools such as SnortSnarf, ACID and RazorBack are described in [1] and
suffer from similar limitations as well as a heavier reliance on text-based presentation of information. Real attacks will generate many alarms [4], rapidly filling logs with redundant information, until stopped. This fact, together with the average amount of unique alarms generated, can cause information overload and possibly hide the most significant attacks. To address the problem of oppressively large alert logs, IDS RainStorm provides security analysts and network administrators with an informative, information-rich display and a convenient interface for monitoring network security alarms. What differentiates our IDS RainStorm from these others is that 2.5 class B IP address spaces (65,532 hosts × 2.5 = 163,830 total) can be represented successfully on one display. Mapping alarms to pixels encodes a large amount of alarm data into one screen for a full 24 hour period.

In the research domain there have been several innovative approaches that seek to overcome the problem of analyst overload by visualizing network packets. Current research systems employ scatterplots, parallel coordinate plots, line segments, glyphs, maps, graphs and similar high-level techniques. Unfortunately, the great majority of these systems depend heavily on a small subset of packet header fields, such as source and destination network addresses and ports, but neglect lesser used fields and the application layer payload. In the current security environment, a large percentage of malicious activity occurs using the less common header fields and in the application layer. As a result, such systems are effectively blind to these classes of attack. To address these issues, our second system, RUMINT extends several best practice visualization techniques by incorporating 19 header fields, an interactive personal video recorder (PVR) metaphor, and a binary rainfall visualization that allows comparison of 600-1000+ payloads at one time. Our results indicate that the RUMINT system presents, in a usable and effective manner, 45-360 times more information than traditional hexadecimal representations.

3 A DESIGN FRAMEWORK FOR SECURITY VISUALIZATION SYSTEMS

In this section we propose a framework for the design of security visualization systems (Figure 2). While system frameworks are difficult to validate categorically, our proposal emerged from several years of directed research. During this period, we iteratively designed and implemented six security visualization systems and conducted an extensive survey of commercial, open source and research systems. From this review, we noted distinct similarities in the architecture and processing pipelines of many systems, but were unable to find an underlying framework in the literature to inform our designs. We also noted that several systems advanced valuable, but rarely seen, components that we believe other researchers should consider. Finally, we received feedback from users on several key areas that were lacking from any of today’s systems. By merging insights from all of these sources we have attempted to create a comprehensive framework. We believe that by better defining the components and processing sequence of security visualization systems other designers will more rapidly be able to design and construct effective systems. In addition, by closely examining each individual component in isolation there lies great potential for future work and optimization. We also believe that the lessons learned, which are embedded in the framework, will assist researchers working in other domains, particularly those constructing interactive information visualization systems. The following sections describe the major components of the framework and illustrate its use by decomposing the Ethereal system. It is important to note that not all components of the framework must be implemented for a successful system, but we believe designers should, at least, consider each stage.

inputs: Possible inputs to the system may take many forms across a broad spectrum of data quality, from unprocessed data to highly refined semantic information. For example, Snort performs signature matching against network traffic to provide specific alert information. Ethereal collects only raw capture data from network packets. Sources may include flows from security sensors, firewalls, intrusion detection systems, network servers, host-based security sub-systems and honeynets. Inputs are not constrained to these, typically passive, traditional sources. Additional semantic information may be infused into the visualization system by including active collection flows such as those provided by the nmap network mapping tool as well as more specialized tools, such as the p0f passive operating system fingerprinting tool. Single streams of security flows are used in most implemented systems; effective integration of multiple streams to support improved correlation remains an open problem. Timeliness of the data will range from real-time, near-real-time and historical information. It may be collected directly by the visualization system, received from external devices or pulled from intermediate databases.
**preprocessing and pre-filtering:**
The data and information flows received by the system may or may not be in a format compatible with the system. In many instances they will need to be parsed and relevant information will need to be extracted. Ethereal does this comprehensively through the use of dissectors for 706 different protocols. Pre-filtering allows users to select only the desired subset of records/fields to progress further up the processing pipeline in an effort to conserve system resources. Ethereal implements this capability through the use of a capture filtering language.

**system storage (RAM Cache & Disk Buffer):** After preprocessing and pre-filtering, the data may be buffered. The buffer typically consists of a RAM cache and may include fall over storage on disk for large data streams. Ethereal behaves in this manner. Such storage is optional and may be bypassed in instances where interactivity is at a minimum and state is not required.

**post filtering and encoding:** Filtering and encoding are logically intertwined. Before being passed to the graphics engine and subsequent visualization, the user makes choices based on what information they would like to view and how to display it. Ethereal uses a display filter language to filter data in the buffer and provides a coloring capability to encode additional information in the display.

**graphics engine and visualizations:** The graphics engine receives the remaining components of the data flows as well as encoding instructions and passes the information to the visualization displays. The visualizations display the information using a variety of information visualization techniques and may include any number of semantic windows on the data. Typically these visualizations are graphical in nature, but may exploit other senses such as sound and touch. Ethereal provides a three pane multiple coordinated display which includes an interactive textual list, tree based protocol decodes and the raw hex/ASCII representation of the selected packet. In the future, the graphics engine and visualization windows may include ties to machine processing modules to direct and conserve human attention.

**logging and reporting:** Visual logging and reporting are relatively unexplored aspects of security visualization systems, but our interviews with security analysts indicated that they are quite important, particularly the reporting task, for communicating results to other analysts, end users, customers and managers. Visual logging of security data includes automatically storing images and video clips of visualization activity in lieu of storing the underlying raw source data. Visual reporting exploits the strengths of visualization systems by allowing the analyst to work through slices of network traffic and, once an area of interest is determined, allows easy construction of a summary report that may include marked-up images, video clips, filtering parameters and analyst comments. Ethereal incorporates neither of these capabilities.

![Figure 3: IDS RainStorm main view](image)

**4 IDS RAINSTORM: IDS ALARM DATA PRESENTATION**
In IDS RainStorm alarm data is presented in an overview where system administrators can get a general sense of network activity and easily detect anomalies. Zooming and drilling down for details can be performed at the user’s discretion. The visualization tool, dataset and examples are presented in this section.

**4.1 StealthWatch IDS Alert Flows**
The Stealthwatch [5] anomaly based IDS system is one of the security appliances used to secure the Georgia Tech campus. It monitors flow activity and bandwidth usage to detect anomalous behavior. In order to test our system, we used Stealth Watch IDS alarm logs generated from Internet traffic on the perimeter of the Georgia Tech network. It is important to note that IDS Rainstorm can be used for other IDS system alarm logs as well. An average of 7,000 alarms are generated in one day from StealthWatch.
The StealthWatch IDS contains two alarm parameters and 33 alarm types that we use in our visualization tool. When each alarm is generated, a Unix timestamp is recorded. Time helps determine an alarm’s temporal position among the rest of the alarms and can help to find significant patterns or position in a sequence of events. Finally, the associated victim IP address and any external IP source address is given.

4.2 Visualization System

IDS Rainstorm provides a main view that presents an overall representation of the entire Georgia Tech IP range and a zoom view that provides more information on a user-selected range of IP addresses. The overall view was designed to convey enough information so that an administrator can see network activity that needs immediate attention. Once alerted to patterns of suspicious network activity, administrators can retrieve specific details of particular alarms using the zoom view.

4.2.1 Visual Representation and Main View

Each of the views follows a general visualization technique developed to address this problem as shown in Figure 3. The visualization uses a set of rectangular regions that represent (top-to-bottom) the set of contiguous IP addresses, where 20 addresses are allocated to a row of pixels. Each column’s horizontal width represents 24 hours of network activity. Individual colored dots in a row (IP addresses) represent total alarms for those 20 addresses at a particular point in time (horizontal position). The alarm with the most severity out of the 20 addresses will appear. Color represents alarm severity where red is high concern, yellow is medium concern, and green is low concern.

The parameter with the largest range of values, therefore the largest scaling problem, is the 2.5 Class B IP addresses. Since a way is needed to show an overview of all of them without cluttering the view, we applied a method used in the Tarantula tool [6] and the SeeSoft tool [7] for representing large source code files. Each represents a source line as a line of pixels, and then simply wraps around to the next column to continue the sequence of source lines. Scaling time is a less taxing problem since its range is variable. We use 24 hours for the range shown in detail in Figure 3 since alarm logs are generated for every 24 hours by default. Scaling 24 hours onto the total width of a typical screen resolution divided by the number of y axes worked well without highly cluttered pixels in the main view. Each pixel on the x-axis represents 20 minutes, and each IP on the multiple y-axes represent approximately 20 IP addresses.

4.2.2 Zoom View

As a user moves the mouse across the overview, a red box highlights the current cursor position as illustrated in Figure 3. This red box is an IP range selector and prints the IP address in the top position. When a user clicks on the overview, a secondary screen appears in a separate window with an enlarged view of the portion enclosed by the red box. Labels are on the top horizontal axis to represent time within 24 hours. Alarms are represented as larger glyphs as seen in Figure 4.

The extra space in the zoom view is used to provide other information such as additional alarm detail for each alarm and external IP address connections. External connections are shown with lines pointing to the affected internal IP address. We implemented an additional zoom function, based on time and local IP, within the zoom view when the mouse is double clicked. Within this same window, the layout is redrawn (Figure 5). Zooming is helpful in reducing overlap when more than one alarm occurs for an IP address at the same time, and for addresses that are close together in position.

4.2.3 Other Techniques

Glossing: Glossing occurs when a user moves the mouse cursor over an icon or particular text, and expanded information is presented. In the zoom view, when a user mouses over a particular alarm glyph, a pop-up gloss is shown that gives the alarm type, time, source and destination IP address. Also, mousing over an external IP

Figure 4: Botnet activity shown in a zoom view for the same IP address space on April 26, 2005 (left) and April 27, 2005 (right). Internal IP addresses are on the left vertical axis and external IP addresses are on the right vertical axis. The activity time pattern for the two days is almost identical.

Figure 5: Worm activity for a particular host located in the campus dorms. The left side is the zoom view and the right side is a 2x zoom that focuses on activity from 3:00-6:00pm.
creates a gloss, highlights the respective address and triggers the plotting of a line which connects the external IP address to the alarm glyph on the graph. This is useful when multiple external IP addresses overlap in the same area on the left axis, making it harder to read. Examples of these methods are shown in Figure 5.

**Filtering**: In both the overview and zoom views, the user may filter on alarm severity, choosing to show only the critical alarms (red), medium concern alarms (yellow), or the low concern alarms (green). This capability can help the user focus on particular alarms for further analysis and to sort through multiple alarms that appear at the same time for a given set of IP addresses.

A cluster of red alarms shown in region 2 of figure 6 can be seen in the midst of more common medium priority alarms in the dorm IP space (region 1 in Figure 6). Here one IP address was a source for StealthWatch’s *Watch Port Active* alarms (which indicate that a port on the user defined watch list has become active) from many external IPs as shown in Figure 5. A close-up of this activity showing the 3:00-6:00 PM time range can also be seen in Figure 5. The infected host demonstrated characteristic communication to a wide range of IP addresses as seen by the line of glyphs and array of line segments connecting to external IP addresses.

On the same day there is another cluster of red alarms (region 3, Figure 6). These alarms are *Watch Host Active* (indicates that a host on a user specified watch list has become active). Some of these external hosts have made connections to other hosts on the local network previously and had bots installed on them, which is why they were placed on the list. These bots were more active around midnight and in the figure we can see similar activity around midnight. The next day, for the same IPs, you can see almost the same time pattern of activity. The zooms for each consecutive day can be seen in Figure 4. We can conclude that these IP addresses have become infected with a bot, which has a specific time pattern of activity.

These examples show how analysis is improved for reoccurring alarms, due to general dorm activity, and ones that were triggered due to anomalous behavior (botnet and worm case). The visualization enhances the analysts’ view of the logs and allows them to more easily notice activity that machines cannot. Other monitoring tools can optionally be re-coded or re-calibrated according to insights gained from human observation. Nonetheless, the tool is only good as the data it receives; therefore, some problems can be difficult to find especially when false alarms are part of the data.

![Figure 6: Overview of April 27, 2005 alarms. (Two regions are artificially identified in green and magnified for easier viewing.) Here region 1 shows activity in a subset of campus dorm IP addresses, a cluster of activity for a machine in the dorm is outlined in region 2 and region 3 shows a cluster of activity occurring over a small range of IP addresses for the entire day.](image)
These visual images can give a system administrator a frame of reference of what a usual day looks like. If any day deviates from this image, then the system administrator may need to investigate further to find out if the change is anomalous or not. Comparing a new view to a normal day's image is a much faster process than trying to do the same with text logs (the image of a day can be saved for later reference). This capability is quite significant given the amount of traffic that a large campus or enterprise generates. This type of analysis also shows the advantage a human has over machine learning algorithms used to find anomalous activity.

4.4 Results

When we demonstrated a prototype version of IDS Rainstorm to system administrators, they uniformly gave positive feedback and clearly indicated that they needed help with the log analysis task. Some of the suggestions they made were the following:

- Remap the two axes such that the entire internal IP address range is on the left and a small set of suspicious external IPs are on the right. For example, if a worm is targeting a network and the IPs affected are spread across the IP space of the network, then it is harder to correlate the behavior. A subset of these external IPs that are connecting to the local network can be plotted on the right parallel axis and the entire local IP space condensed on the left axis. This will help to see what hosts are triggering alarms due to activity of the external IP address.

- Combine alert outputs from the other IDS systems used in the tool to compare each system's output and help rule out false alarms.

Currently, IDS Rainstorm is useful for visualizing IDS alarms on a large network, observing time patterns, knowing location (local and external IPs) and severity. Our analysis of the requirements and tasks of the system administrators of Georgia Tech's network identified that these capabilities would be helpful. The tool presently can be used for forensic analysis, but we also would like to implement real-time analysis for live monitoring of the network. In order for the tool to be used on the network, system administrators will have to learn how to use it, how to interpret the display and what the visual patterns mean. People are generally good at these tasks and we are optimistic that system administrators will grasp these concepts quickly.

5 RUMINT: A PVR APPROACH TO PACKET-LEVEL VISUALIZATION

5.1 Design Overview

The primary design goal of RUMINT is to provide users the ability to view a large number of network packets in a way that supports rapid comparison, deep and broad semantic understanding, and highly efficient analysis. At the same time, we wish to allow intuitive interaction in order to remove noise and highlight packets of interest. It consists of a PVR interface and seven primary visualizations (see Figure 7); each designed to provide different semantic windows on network traffic. These include:

thumbnail toolbar (Figure 7a): The thumbnail toolbar provides a real-time overview of each visualization window in a thumbnail size display. Doubling as a menu, users may bring up the full size window by clicking on a thumbnail.
scrolling text display (Figure 7b): The scrolling text display presents network packets, one per horizontal row, in a user selectable encoding (ASCII, hexadecimal and decimal). It includes a strings command, adopted from the Unix environment, that will filter based on packet contents and display only sequences of characters from the printable ASCII range (e.g. strings of length 3-9).

parallel coordinate plot display (Figure 7c): This visualization uses the parallel coordinate plot technique to display scaled values from packet header fields. Currently 19 header fields, up to 19 vertical axes and 19! combinations of headers are supported.

detail display (Figure 7d): The detail window displays the selected packet’s contents in a traditional hex/ASCII format.

glyph-based animation display (Figure 7e): The glyph-based display combines three display planes to animate any two attributes (header fields) of network traffic. The center pane is a two axis parallel coordinate plot and the side planes contain glyphs which move off the screen as the network traffic is processed.

binary rainfall visualization (Figure 7f): The binary rainfall visualization displays packet contents, one per line. It has three primary views which map packet contents to display pixels.

scatterplot display (Figure 7g): The scatter plot visualization allows users to select any two header fields (19 are implemented) and plots them on a traditional X,Y display. Header field values are scaled to match the dimensions of the display window.

byte frequency display (Figure 7i): The byte frequency visualization displays the presence and frequency of bytes within each packet.

PVR interface (Figure 7h): The PVR interface is the heart of the RUMINT application. Packets are captured live from the network or loaded from capture files and stored in an internal cache. The PVR interface allows playback of these packets for viewing in any of the visualization windows. This approach extends the VCR metaphor suggested by Erbacher [8].

From the near infinite space of possible visualization techniques these seven were chosen based on both intuition and feedback from security analysts. During the design process we explored approximately 15 other visualizations, but these were ultimately discarded because they did not most effectively address user needs. We believe these seven visualizations represent a solid set of techniques that can be refined to increase their utility, but it is important to note that the PVR based design of the system scales well. It is a straightforward matter to add new visualizations in a very short time period.

In sections 5.3 and 5.4 we highlight two of these visualization techniques, the binary rainfall and the byte frequency display, and examine their utility in a case study of malicious traffic in section 5.5.

5.2 Binary Rainfall Visualization
The binary rainfall visualization (Figures 8 and 9) was inspired by the classic waterfall display used for spectrum analysis but instead plots packets, one per horizontal line, in time sequence order. There are three graphical views
which plot pixels in direct correspondence to the structure of the binary data. These views include plotting each bit of binary data as a monochrome pixel, each byte of binary data as a 256-level grayscale pixel and each three bytes of binary data as 24-bit RGB pixel. The primary benefit of this visualization is its ability to rapidly compare up to 1,000 packets. Textual approaches, such as those found in Ethereal, are limited to about 40 packets per screen. Other benefits include the ability to compare packet lengths, identify identical values between packets and to support signature development for network based malicious software.

5.3 Byte Frequency Visualization

The byte frequency visualization (Figures 8 and 9) employs a similar approach by plotting one packet per horizontal line. Pixels along the horizontal axis, scaled from 0-255, are illuminated based upon the frequency with which the corresponding byte appears relative to each packet. The pixel may be illuminated as a single color if one or more of a given byte is present (byte presence) or encoded with color based upon the frequency (byte frequency). The key strength of this visualization is its ability to facilitate rapid comparison of packet structure and contents including such applications as detecting the use of encryption, fingerprinting executable files, detection of ASCII text and analysis of polymorphic network worms.

5.4 Honeynet Case Study

Operationally, we used the RUMINT system to monitor Georgia Tech Honeynet traffic for twelve months, from July 2004 to June 2005. In a typical usage scenario, users loaded datasets of interest and iteratively adjusted the menu parameters to focus on areas of interest. For example, a user examining a honeynet dataset wished to filter as much Internet background radiation [9] as possible. Being familiar with Pang’s observation that a portion of UDP traffic is caused by messenger spam, the user wished to perform the following actions: constrain the visualization to display only UDP traffic from common messenger ports, confirm that the traffic was indeed messenger spam and finally to filter those packets from the larger dataset. The user first viewed the entire data set and noted that a portion of the traffic contained groups of nearly identical packets (Figure 8 left) with a high percentage of bytes in the printable ASCII range (Figure 8 right). Using the scrolling text display, the user confirmed the traffic as messenger spam (Figure 10) and created a filter for use with future datasets.

5.5 Results

The RUMINT system has been deployed globally in operational, laboratory and training environments and has over ten thousand copies in circulation. Users report significantly improved analysis on datasets of up to 100,000 packets. We have engaged several hundred users directly through focus groups, interviews and demonstrations. Through this interaction we found overwhelming support for the PVR metaphor. We also found that our user base liked the notion of visually examining large slices of network traffic and then using Ethereal to examine a small number of identified packets. This complementary approach exploits the strengths of both systems. Visualization provides big picture context and helps direct the analyst to areas of interest. Ethereal excels at protocol dissection and analysis of a small number of packets. The end result is a significant gain in analyst performance far exceeding the current limitations of 6,631 packets, as found in our study of analysts, using Ethereal alone. Beyond the current ceiling of 100,000 packets, the system does depend on machine processing technologies to help identify interesting groups of packets to pass to the visualization system. As an interim measure, we have examined datasets that have limited amounts of legitimate traffic: honeynets, capture the flag exercises and botnets. We plan to explore such automated support for the system in future work.
6 CONCLUSIONS

Information visualization of security related data bears great promise in making our personal computers, servers and networks more secure. Such work is both an art and a science requiring expertise from the computer graphics, information visualization, interface design and security communities in order to turn the raw security data into insightful and actionable information and knowledge. There is no shortage of raw data, in fact there is far more than can be analyzed by today’s best tools. Humans often cope with this torrent of data by using crude statistical techniques, textual displays, outdated graphical techniques and by ignoring large portions of it. We believe that security visualization, at its best, is both compelling as a video game and several orders of magnitude more effective than the tools we employ today. In this article, we moved toward this goal by exploring the design, implementation and evaluation of two prototype systems springing from immediate, high priority security needs and developed by an interdisciplinary team of researchers. By bringing together diverse ideas and expertise we directly addressed significant problems facing the people who defend our information technology resources.

BIBLIOGRAPHY


