Countering Security Information Overload through Alert and Packet Visualization

Gregory Conti, Kulsoom Abdullah, Julian Grizzard, John Stasko, John A. Copeland, Mustaque Ahamad, Henry L. Owen, and Chris Lee
Georgia Institute of Technology

This article presents a framework for designing network security visualization systems as well as results from the end-to-end design and implementation of two highly interactive systems.

The massive amount of security data that network sensors and host-based applications generate can quickly overwhelm the operators charged with defending the network. Often, operators overlook important details and it's 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 operators' 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 complementary security visualization systems. The first system, intrusion detection system (IDS) RainStorm (see Figure 1), provides high-level overviews of intrusion-detection alerts. The second system, Rumint, provides detailed insights into packet-level network traffic. These systems mirror two primary tasks of security analysts: detect and respond to network intrusions (IDS RainStorm) and perform rapid in-depth analysis of specific intrusion events (Rumint).

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 results of our survey of security professionals, the design framework, lessons learned from the design of our systems as well as an evaluation of their effectiveness. Our results indicate that both systems effectively present significantly more information when compared to traditional textual approaches. We believe that the interactive, graphical techniques that we present will have broad applications in other domains seeking to deal with information overload. This article is based on a series of conference and workshop papers that describe earlier versions of our work.[1,2]

Evaluating current best practices

Information overload is an everyday occurrence for security analysts. While there is a tremendous amount of work on information overload by the psychology research community, little has been done that directly examines the real-world needs of security professionals and network administrators. We believe this specificity is critical to developing effective solutions.

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 to 35,000 networked computers operational at any given time. The total number of 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 600 megabits per second. On average, the network processes more than 4 terabytes of data each day.

Georgia Tech's Office of Information Technology manages the security, health, and welfare of the campus network. Staffed by a handful of network analysts and

 Published by the IEEE Computer Society

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. 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, which have proved 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.

Surveying the limits of current tools

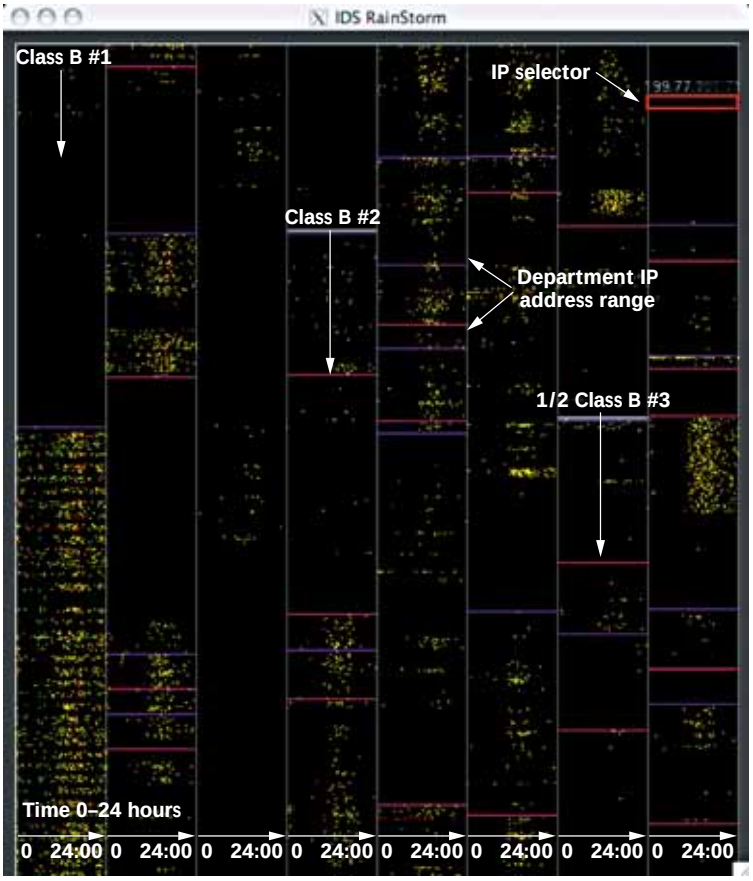
The problem is larger than just the Georgia Tech campus. As part of our research, we surveyed 39 professional security analysts working in industry and graduate students specializing in information security. Our survey studied the limits of two widely used open source tools: Snort, an IDS, and Ethereal, a protocol and packet analysis tool.

Our results indicated 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. Information-security students had a lower capability to cope with alerts, reaching overload at an average of 30 alerts per hour. Professional security analysts could handle more alerts than students, but most still faced overload at an average of 230 alerts per hour. The situation is similar for packet analysis. Information-security students felt that Ethereal became difficult to use when the number of packets exceeded approximately 500 packets. Professional security analysts handled a larger number of packets without overload, but most respondents from this group felt that Ethereal became difficult to use when faced with data sets exceeding 6,600 packets.

Causes of overload

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

For those who had used Snort (15 participants), 40 percent found analysis and ease of use difficult, particularly the graphical front ends available for the tool. Sixty percent were concerned that intrusion-detection signatures were easy to bypass and difficult to tune effectively for a given network. Of these, 56 percent felt that an attacker could easily bypass existing signatures and 44 percent found it difficult to effectively tune the intrusion-detection system to minimize false alarms and missed attacks.

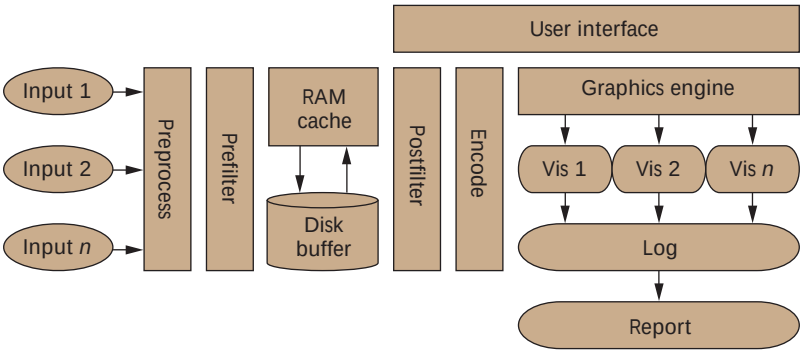


1 IDS RainStorm main view: The eight vertical axes represent the 2.5 class B IP addresses. The thicker horizontal lines between these axes show where each class B starts. Other horizontal lines show the start and end of each department. This screen shot shows an entire day's worth of real alarms. The IP selector box is also notated (at the top right).

For those who had used Ethereal (27 participants), 59 percent indicated problems with the ability to see the big picture in the data due to overwhelming detail. Forty-four percent 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. Eighteen percent found the filtering capability difficult to learn and use.

How our systems address overload

The two systems we present in this article directly address the majority of these issues. Note, 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 information overload. Therefore, our systems 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 com-



2 Framework for the design of security visualization systems. Security data flows through a series of intermediate processing, encoding, and filtering steps before being visualized. After visualization, a system can log the results graphically and create custom reports.

pared 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 shows IP address connections over time. This tool is successful in combining multiple parameters, visually representing them to assist analysts in finding anomalous behavior. However, the amount of information shown is limited to a subset of IP address ranges, time (4 hours), and number of attacks. We describe other related tools such as SnortSnarf, ACID, and RazorBack elsewhere.[1] These tools 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 the other tools is that it represents 2.5 class B IP address spaces (65,532 hosts × 2.5 = 163,830 total) 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, several innovative approaches have sought 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, Rumint extends several best practice visualization techniques by incorporating 19 header fields, an interactive personal video recorder metaphor, and a binary rainfall visualization that allows comparison of 600 to 1,000 or more payloads at one time. Our results indicate that the Rumint system presents, in a usable and effective manner, significantly more information than traditional hexadecimal representations.

Design framework for security visualization systems

We propose a framework for the design of security visualization systems (see Figure 2). While system frameworks are difficult to validate categorically, our proposal emerged from several years of 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 be able to design and construct effective systems more rapidly. In addition, closely examining each individual component in isolation gives us 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. Note, 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 might 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 might include flows from security sensors, firewalls, IDSs, network servers, host-based security subsystems, and honeynets. (Honeynets are computers placed on a network, but have no legitimate use, therefore all activity is suspect.) Inputs are not constrained to these, typically passive, traditional sources. Additional semantic information might 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. Most implemented

systems use single streams of security flows; 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 might be collected directly by the visualization system, receive from external devices, or pulled from intermediate databases.

Preprocessing and prefiltering

The data and information flows the system receives might or might 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. Prefiltering lets users select only the desired subset of records or 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

After preprocessing and prefiltering, the data might be buffered. The buffer typically consists of a RAM cache and might include additional storage on disk for large data streams. Ethereal behaves in this manner. Such storage is optional and might be bypassed in instances where interactivity is at a minimum and state is not required.

Postfiltering and encoding

Filtering and encoding are logically intertwined. Before being passed to the graphics engine and subsequent visualization, users make 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 might include any number of semantic windows on the data. Typically these visualizations are graphical in nature, but might exploit other senses such as sound and touch. Ethereal provides a three-pane multiple coordinated display that 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 might 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 might include marked-up images, video clips, filtering parameters, and analyst comments. Ethereal incorporates neither of these capabilities.

By better defining the components and processing sequence of security visualization systems, other designers will be able to design and construct effective systems more rapidly.

IDS RainStorm

IDS RainStorm presents alarm data 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. After using IDS RainStorm to rapidly identify events of interest, system administrators can then examine the associated network packets using Rumint for more detailed analysis.

StealthWatch IDS alert flows

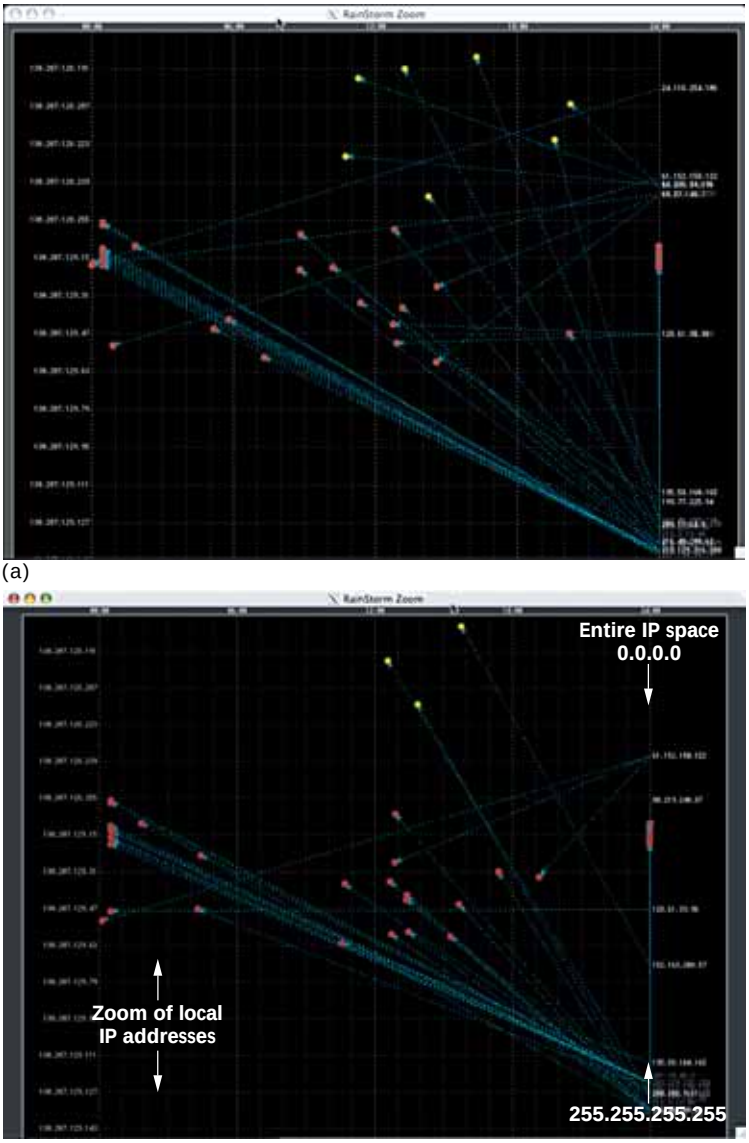
Lancope's StealthWatch anomaly-based IDS system is one of the security appliances used to secure the Georgia Tech campus (see http://www.lancope.com/products/). It monitors flow activity and bandwidth usage to detect anomalous behavior. To test our system, we used StealthWatch IDS alarm logs generated from Internet traffic on the perimeter of the Georgia Tech network. IDS RainStorm can be used for other IDS system alarm logs as well. StealthWatch generates an average of 7,000 alarms in one day.

The StealthWatch IDS contains two alarm parameters and 33 alarm types that we use in our visualization tool. After the system generates an alarm, it records a Unix time stamp. Finally, the tool uses the associated victim IP address and any external IP source address provided by StealthWatch.

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. We designed the overall view to convey enough information for an administrator to 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.

Visual representation and main view. Each of the views follows a general visualization technique developed to address this problem, as Figure 1 shows.



3 Botnet activity shown in a zoom view for the same IP address space on (a) 26 April 2005 and (b) 27 April 2005. (Botnets are groups of compromised computers that are under the command and control of a malicious entity.) Internal IP addresses appear 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.

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 highest severity out of the 20 addresses will appear onscreen. In addition, the user has the option of configuring StealthWatch to correlate a series of low-priority events into a single higher priority alarm to reduce visual clutter. 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. Because users need a way see an overview of all addresses without cluttering the view, we applied a method used in the Tarantula[5] and SeeSoft tools[6] 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 because its range is variable. We use 24 hours for the range shown in detail in Figure 1 since alarm logs are generated 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 pixel on the multiple *y*-axes represents approximately 20 IP addresses.

Zoom view. As a user moves the mouse across the overview, a red box highlights the current cursor position as illustrated in Figure 1. 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. IDS RainStorm represents alarms as larger glyphs, as Figure 3 shows.

The extra space in the zoom view provides other information such as additional detail for each alarm and external IP address connections. In Figure 3, external connections appear with lines pointing to the affected internal IP address. We implemented an additional zoom function, based on time and local IP, that users can double click the mouse to pull up within the zoom view. Within this same window, the system redraws the layout (see Figure 4). Zooming is helpful in reducing overlap when more than one alarm occurs for an IP address at the same time, and for addresses that appear close together.

Glossing. Glossing occurs when a user moves the mouse cursor over an icon or particular text and the program presents additional information. In the zoom view, when a user mouses over a particular alarm glyph, a pop-up gloss appears that gives the alarm type, time, source, and destination IP address. Also, mousing over an external IP creates a gloss, highlights the respective address, and triggers the plotting of a line that 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 difficult to read. Figure 4 shows an example of this method.

Filtering. In both the overview and zoom views, the user might filter on alarm severity, choosing to show only the critical alarms (red), medium-concern alarms (yellow), or the low-concern alarms (green), as shown in Figure 4. 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.

Example usage scenarios

Certain attacks, like botnet and worm activity, can target a network across the logical IP space from one or a range of source IP addresses. When this happens, the attack is not as obvious to the network adminstrator until an abundance of alarms are generated (which happens when many machines have been targeted) or the effects of the attack become more dominant across the network **//Okay?//**. A useful activity is to pan through the graph by clicking the mouse and dragging the IP range selector, or red box, through the overview. The resulting motion appears in the zoom view (see Figure 4a). Time is constant while the internal IP addresses on the left vertical axis change sequentially. The external IP addresses on the right axis maintain the same 2^{32}-bit mapping but as the user scrolls in the main view, the external IP addresses appear (and disappear) based on alarm activity associated with the changing/moving internal IP addresses. This activity allows traversal through the range of IP addresses to find detailed patterns. The external IP addresses remain constant through the panning, and this helps determine whether there is some address or range of addresses trying to attack the network.

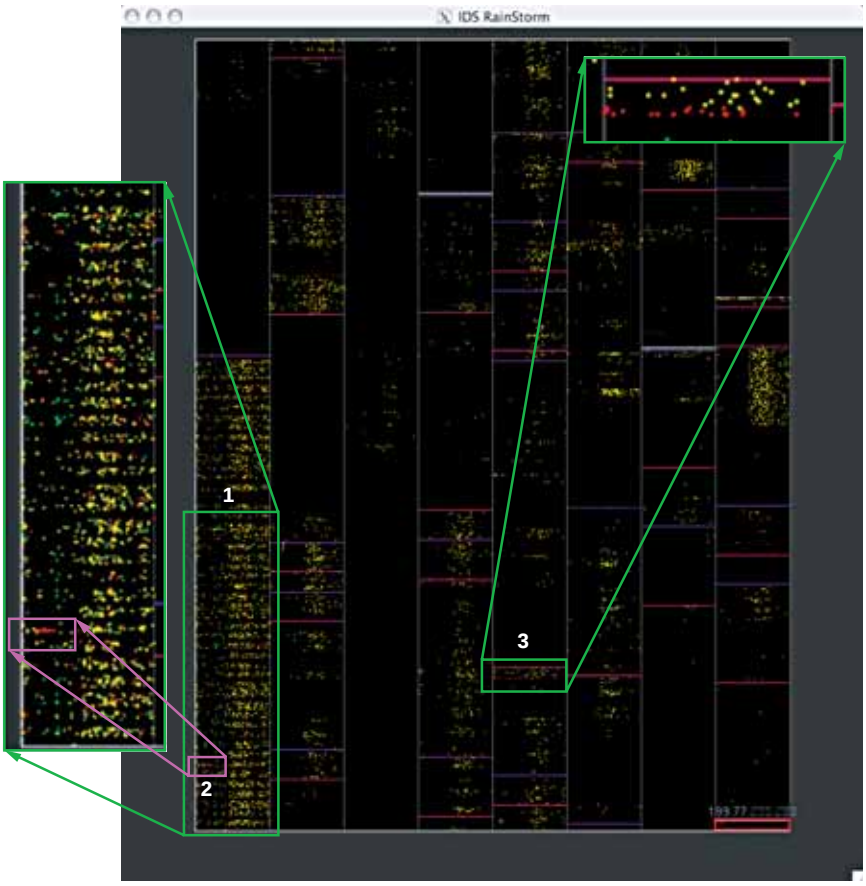
A cluster of red alarms in region 2 of Figure 5 (next page) is easily visible in the midst of more common medium priority alarms in the Georgia Tech campus dorm IP space (region 1 in Figure 5). Here, one IP address was a source for StealthWatch's Watch Port Active alarms (which indicate that a port on the user-defined watch list has become active) from many external IPs as Figure 4 shows. The figure also highlights a close-up of this activity showing the 3:00 p.m. to 6:00 p.m. time range (see Figure 4b). The infected host demonstrated characteristic communication to a wide range of IP addresses as 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 5). 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. Figure 3 shows the zooms for each consecutive day. 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 for alarms that were triggered due to anomalous behavior (botnet and worm case). The visualization enhances the analysts' view of the logs and lets them more easily notice activity that machines cannot. Other monitoring tools can optionally be recoded or recalibrated according to insights gained from human observation. Nonetheless, the tool is only as good as the data it receives; therefore, some problems can be difficult to find especially when false alarms are part of the data. The underlying IDS system generates this alarm data and, unfortunately, even today's best systems are prone to some degree of false alarms and are unable to detect all classes of anomalous activity. IDS RainStorm implements general techniques for displaying whatever alarms are generated by any IDS.

4 Worm activity for a particular host located in the Georgia Tech campus dorms: (a) Zoom view and (b) a 2x zoom that focuses on activity from 3:00 p.m. to 6:00 p.m.

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 might need to investigate further to find out whether the change is anomalous. 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.



5 Overview of alarms on 27 April 2005. (Two regions are artificially identified in green and magnified for easier viewing.) Region 1 shows activity in a subset of campus dorm IP addresses. Region 2 outlines a cluster of activity for a machine in the dorm. Region 3 shows a cluster of activity occurring over a small range of IP addresses for the entire day.

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 so that the entire internal IP address range is on the left and a small set of suspicious external IPs are on the right.
- Combine alert outputs from the other IDS systems used in the tool to compare each system's output and help rule out false alarms.

In the case of remapping the two axes, 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 difficult to correlate the behavior. A subset of these external IPs that connect 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.

Currently, IDS RainStorm is useful for visualizing IDS alarms on a large network, observing time patterns, knowing locations (local and external IPs), and severity. Our analysis of the requirements and tasks of Georgia Tech's network system administrators 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. We also wish to explore how well the tool will scale if the amount of alarms were to multiply. We need additional testing in this area, and we predict that more user interaction as well as enhanced filtering and improved querying capabilities will be required to compensate for the increased number of alarms.

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.

Rumint

The primary design goal of Rumint is letting users view a large number of network packets in a way that supports rapid comparison, deep and broad semantic understanding, and highly efficient analysis. We also aimed to allow intuitive interaction to remove noise and highlight packets of interest. We purposely designed Rumint to complement higher level systems such as IDS RainStorm. IDS RainStorm excels at identification of events of interest, but lacks comprehensive tools for analyzing the underlying network packets that caused the event. We created Rumint to fill this gap.

Rumint consists of the following seven visualizations (in addition to personal video recorder interface), each designed to provide different semantic windows on network traffic:

- scrolling text display,
- parallel coordinate plot display,
- glyph-based animation display,
- thumbnail toolbar,
- binary rainfall visualization,
- byte frequency display, and
- detail display.

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 (for example, strings of three to nine characters).

The parallel coordinate-plot display visualization uses the parallel coordinate-plot technique to display scaled values from packet header fields (see Figure 6a). Cur-

rently, the display supports 19 header fields, up to 19 vertical axes, and 19! combinations of headers.

The glyph-based display combines three display panes to animate any two attributes (header fields) of network traffic (see Figure 6b). The center pane is a two-axis parallel coordinate plot and the side panes contain glyphs that move off the screen as the network traffic is processed.

The thumbnail toolbar provides a real-time overview of each visualization window in a thumbnail-size display (see Figure 6c). Doubling as a menu, users may bring up the full size window by clicking on a thumbnail.

The binary rainfall visualization displays packet contents, one per line (see Figure 7a on next page). It has three primary views that map packet contents to display pixels. The byte frequency visualization displays the presence and frequency of bytes within each packet (see Figure 7b on next page). A detail window (not shown) displays the selected packet's contents in a traditional hex/ASCII format.

From the near infinite space of possible visualization techniques, we chose these seven based on 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 effectively address user needs. We believe these seven visualizations represent a solid set of techniques that we can refine to increase their utility.

The personal video recorder interface is the center of the Rumint application (see Figure 6d). Packets are captured live from the network or loaded from capture files and stored in an internal cache. This interface allows playback of these packets for viewing in any of the visualization windows. This approach extends the videocassette recorder (VCR) metaphor suggested by Erbacher.[7] In addition, the personal video recorder-based system design scales well. It's a straightforward matter to add new visualizations in a short time period.

Binary rainfall visualization

The binary rainfall visualization (see Figure 7a) was inspired by the classic waterfall display used for spectrum analysis but instead plots packets, one per horizontal line, in time-sequence order. Three graphical views plot pixels in direct correspondence to the binary data's structure. These views include plotting each bit of binary data as a monochrome pixel, each byte of binary data as a 256-level gray-scale pixel, and each three bytes of binary data as a 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 support signature development for network-based malicious software.

Byte frequency visualization

The byte frequency visualization (see Figure 7b) employs a similar approach by plotting one packet per horizontal line. There are 256 positions along the horizontal axis which correspond to the presence or absence

6 Overview of the Rumint packet visualization system. Each window provides specialized insight into network traffic and can be used alone or in combination. (a) Parallel coordinate-plot display, (b) glyph-based animation display, (c) thumbnail toolbar, and (d) personal video recorder interface.

7 (a) Binary rainfall illuminating pixels in a one-to-one correspondence to packet bits and (b) byte frequency visualizations illuminating pixels along one of 256 vertical columns corresponding to the presence of byte values. Each visualization displays one packet per horizontal row.

of byte values (0–255) in the given packet. Pixels are colored based on the frequency with which the corresponding byte appears relative to each packet. The pixel might be illuminated as a single color if one or more of a given byte is present (byte presence mode) or encoded with color based on the frequency (byte frequency mode). The user uses the drop down menu at the bottom of the window to change modes. 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.

Results

We have deployed the Rumint system globally in operational, laboratory, and training environments. Users report significantly improved analysis on data sets 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 personal video recorder 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 depends on machine-processing technologies to help identify interesting groups of packets to pass to the visualization system. As an interim measure, we have examined data sets that have limited amounts of legitimate traffic: honeynets, capture-the-flag exercises (competitive events where multiple teams attempt to attack other teams' computers while defending their own), and botnets. We plan to explore such automated support for the system in future work.

Rumint suffers from several other limitations. The most significant is its lack of advanced filtering capabilities. We believe this to be a tractable problem, which we can readily address by incorporating appropriate elements of Ethereal's display filtering language in the future. Rumint also suffers from problems in the areas of scaling and labeling. These areas are more problematic as no clear-cut solutions exist. We project that both will be improved by enhanced filtering, but plan to explore best-practices research from the general information visualization and interface design research literature.

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 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, and outdated graphical techniques and by ignoring large portions of the data. 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 complementary 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. ■

Acknowledgments

The views expressed here are those of the authors and do not reflect the official policy or position of the US Military Academy, the Department of the Army, the Department of Defense, or the US Government.

References

1. K. Abdullah, et al., "IDS RainStorm: Visualizing IDS Alarms," *Proc. Visualization for Computer Security* (VizSEC), IEEE CS Press, 2005, pp. 1-10.
2. G. Conti, et al., "Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries," *Proc. Visualization for Computer Security* (VizSEC), IEEE CS Press, 2005, pp. 83-90.
3. H. Koike and K. Ohno. "Snortview: Visualization System of Snort Logs," *Proc. Visualization and Data Mining for Computer Security* (VizSEC/DMSEC), ACM Press, 2004, pp. 143-147.
4. K. Julisch, "Clustering Intrusion Detection Alarms to Support Root Cause Analysis," *ACM Trans. Information and System Security*, Nov. 2003, pp. 443-471.
5. J. Eagan, et al., "Visually Encoding Program Test Information to Find Faults in Software," *Proc. IEEE Symp. Information Visualization*, IEEE CS Press, 2001, pp. 33-36.
6. S.G. Eick, J.L. Steffen, and E.E. Sumner, "Seesoft—A Tool for Visualizing Line Oriented Software Statistics," *IEEE Trans. Software Eng.*, Nov. 1992, pp. 957-968.
7. R.F. Erbacher, K.L. Walker, and D.A. Frincke. "Intrusion and Misuse Detection in Large-Scale Systems," *IEEE Computer Graphics and Applications*, Jan./Feb. 2002, pp. 38-48.

Honeynet Case Study

Operationally, we used the Rumint system to monitor Georgia Institute of Technology honeynet traffic for 12 months, from 1 July 2004 to 30 June 2005. In a typical usage scenario, users loaded data sets of interest and iteratively adjusted the menu parameters to focus on areas of interest. For example, a user examining a honeynet data set wished to filter as much Internet background radiation[1] as possible. Being familiar with Pang's observation that a portion of User Datagram Protocol 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 filter those packets from the larger data set. The user first viewed the entire data set and noted that a portion of the traffic contained groups of nearly identical packets (see Figure 7a in the main article) with a high percentage of bytes in the printable ASCII range (see Figure 7b). Using the scrolling text display, the user confirmed the traffic as messenger spam and created a filter for use with future data sets.

Reference

1. R. Pang, et al., "Characteristics of Internet Background Radiation," *Proc. ACM SIGCOMM Internet Measurement Conf.* (ACM-IMC), ACM Press, 2003, pp. 27-40.

Gregory Conti *is a PhD student in the College of Computing at the Georgia Institute of Technology. His research interests include information visualization for security, information warfare and usable security. He has a BS in Computer Science from the United States Military Academy and an MS in Computer Science from Johns Hopkins University. Contact him at conti@acm.org.*

Kulsoom Abdullah *is a graduate research assistant in the Communications Systems Center at the Georgia Institute of Technology, where she is completing her PhD. Abdullah has a BS in computer engineering from the University of Central Florida and an MS in electrical engineering at the Georgia Institute of Technology. Her research interests include network security and visualization. Contact her at kulsoom@ece.gatech.edu.*

Julian Grizzard *is a PhD candidate in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include operating system security, network security, and visualization. Grizzard has a BS in electrical and computer engineering from Clemson University and an MS in electrical and computer engineering from the Georgia Institute of Technology. Contact him at grizzard@ece.gatech.edu.*

John Stasko *is a professor in the College of Computing and the Graphics, Visualization and Usability Center at the Georgia Institute of Technology. His research interests include human–computer interaction and information visualization. Stasko has a BS in mathematics from Bucknell University and an ScM and PhD in computer science from Brown University. Contact him at stasko@cc.gatech.edu.*

John A. Copeland *is the John H. Weitnauer, Jr. Chair at the Georgia Institute of Technology School of Electrical and Computer Engineering. His research interests include information visualization for computer security, network security and high-speed optical networks, Copeland has a BS, MS, and PhD in physics from the Georgia Institute of Technology. He is a Fellow of the IEEE, and received the Morris N. Liebmann award in 1970. Contact him at john.copeland@ece.gatech.edu.*

Mustaque Ahamad *is a professor of computer science in the College of Computing at the Georgia Institute of Technology. His research interests include distributed operating systems, computer security, and fault-tolerant systems. Ahamad has a PhD in computer science from the State University of New York, Stony Brook. Contact him at mustaq@cc.gatech.edu.*

Henry L. Owen *is a professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include network security and network architectures for enhanced security capabilities. Owen has a BS, MS, and PhD in electrical engineering from the Georgia Institute of Technology. Contact him at henry.owen@ece.gatech.edu.*

Chris Lee *is a graduate research assistant in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include security indicator visualization, honeynet monitoring, and wireless optimization at the medium access and routing layers. Lee has a BS in computer engineering and an MS in electrical and computer engineering from the Georgia Institute of Technology. Contact him at chris@ece.gatech.edu.*