Proceedings of the 2005 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY

Real-Time and Forensic Network Data Analysis Using Animated and Coordinated Visualization

Sven Krasser[†], *Member, IEEE*; Gregory Conti[‡], *Member, IEEE;* Julian Grizzard[†], *Member, IEEE;*
Jeff Gribschaw[†], *Member, IEEE;* Henry Owen[†], *Senior Member, IEEE*

Abstract*—Rapidly detecting and classifying malicious activity contained within network traffic is a challenging problem exacerbated by large datasets and functionally limited manual analysis tools. Even on a small network, manual analysis of network traffic is inefficient and extremely time consuming. Current machine processing techniques, while fast, suffer from an unacceptable percentage of false positives and false negatives. To complement both manual and automated analysis of network traffic, we applied information visualization techniques to appropriately and effectively bring the human into the analytic loop. This paper describes the implementation and lessons learned from the creation of a novel network traffic visualization system capable of both real-time and forensic data analysis. Combining the strength of link analysis using parallel coordinate plots with the time-sequence animation of scatter plots, we examine a 2D and 3D coordinated display that provides insight into both legitimate and malicious network activity. Our results indicate that analysts can rapidly examine network traffic and detect anomalies far more quickly than with manual tools.

Index terms—security visualization, network visualization, real-time visualization, forensic visualization, honeynet visualization, honeypot visualization

I. Introduction

The Internet has become one of the nation's critical infrastructures. The large amounts of data transmitted over typical networked systems render it difficult to spot activities by malicious adversaries. This poses a challenge equivalent to "finding the needle in the haystack." Even using high signal to noise ratio (malicious traffic/ legitimate traffic) techniques, such as honeynets or security exercises on isolated networks, analysts are still confronted with large datasets requiring analysis. Packet capture logs from small scale 2-3 day security exercises are on the order of hundreds of megabytes. Even university-level honeynets, which are computer networks with no production value that are set up specifically to watch for network attacks and, by definition, are subject to *only* malicious activity and no legitimate traffic, collect 1-10 megabytes of traffic per day. At one extreme, analysis is conducted using user-customized scripts in parsing languages like Perl to parse the data and produce useful information. At the other extreme, machine processing is used to sift through, consolidate and evaluate data. These machine processing methodologies are highly automated and thus not very adept at recognizing new and unexpected characteristics. In particular, they may abstract away important information without the knowledge of the human operator. A more powerful, and as yet predominately untapped, means of analysis is to use visualization to allow human undirected (getting a general overview about occurrences on the network) as well as directed (displaying and analyzing information regarding a specific incident) processing of the data.

Most security tools generate data so prolifically that they waste the precious resources of human time and attention. As a result, users are unable to efficiently and effectively analyze traffic patterns, easily monitor their networks, rapidly identify attacks, and respond quickly. We believe that carefully crafted tools and application specific visualizations can complement previous approaches by presenting the human with the right amount of information in the right form of presentation at the right time. Our constant battle with information overload during daily analysis activities, in particular with the Georgia Tech Honeynet, motivated us to investigate more human-centric, scalable techniques for the analysis of security data sets. Our primary design goal was to meet this need by designing a visualization system that, when appropriately combined with existing technologies, makes the most effective use of human resources.

The primary contribution of this work is the novel use of tightly-coupled, animated, time-sequence scatter plots and parallel coordinate plots in both 2D and 3D to rapidly analyze network traffic. In addition, we explore the effective use of labeling, animation, scaling, and fading as well as interaction techniques to cope with extremely large ranges of categorical and discrete numeric data. We validate the efficacy of these results by performing analysis of real-time and forensic network traffic from several domains:

- A very active, large-scale university network with two /16 networks and one /17 network.

[†] *School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0250*

[‡] *College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332-0280*

0-7803-9290-6/05/$20.00 ©2005 IEEE.

- Live capture from a university honeynet as well as three years of continuously archived honeynet packet captures.
- Over 80GB of novel capture data from five large-scale botnets collected using a darknet.

Our results indicate that the system provides the capability to rapidly scan large datasets of network traffic for malicious activity despite the visual noise of legitimate, or less important traffic as well as facilitates "at a glance" insight directly supporting network monitoring, intrusion detection and attacker behavior determination.

Section 2 places our work in the field of current research. Section 3 describes our system design and implementation. Section 4 presents our results and evaluation. In sections 5 and 6 we propose areas for future work and present our conclusions. Note that, due to the intensely graphical nature of our work, we suggest that, if possible, you refer to the electronic edition of this document to view the full color images.

II. Related Work

Information visualization is a mature field with a wide range of techniques that have been successfully applied to many domains. See the work by Tufte [1,2,3] and Spence [4] for excellent surveys of the field. But only recently has work been done in earnest to apply these techniques to network security and other related information assurance problems. Examples include Girardin [5] who used self-organizing maps to detect malicious network activity and Nyarko [6] who used haptic feedback to find suspicious activity in network traffic. Another interesting example is the Spinning Cube of Potential Doom [7], which visualizes real-time port and IP data in a three-dimensional cube as a rotating scatter plot. While quite useful to see coarse trends in large-scale networks, it lacks animation, multiple visualizations and interactive capability. The visualization system we outline in this paper combines animated scatter plots with parallel coordinate plots. The notion of parallel coordinate plots was first proposed by Inselberg [8]. Several researchers have applied the technique in the network security domain, including Marchette [9], Conti [10], and the National Center for Advanced Secure Systems with their VisFlowConnect tool [11,12]. We extend their work by combining parallel coordinate views and animated scatter plots into a single cohesive visualization. In addition, we add three-dimensional functionality that allows the user to zoom and pan the combined visualization. We also explore system performance characteristics not seen in other work as well as the use of fading. Fading is used in Etherape, but the tool provides only a single visualization of network traffic arranged in a circular graph [13]. Similarly, Erbacher [14] used a circular visualization composed of animated glyphs to demonstrate that intruder behavior is surprisingly observable. Both of these approaches, while effective for moderate numbers of network nodes, can quickly become crowded. Our combined visualization approach supports a far greater number of nodes. We estimate this gain to be about an order of magnitude more nodes. When combined with our zooming capability, this gain can reach several orders of magnitude, but at the cost of additional human interaction. Our system also extends the work of Conti [10] by adding the ability to capture and display network traffic in real time as well as display historical packet capture files. During the design of our system, we relied upon the work by Komlodi [15] and Fink [16] to ensure we incorporated real-world operator requirements. We also examined best of breed open source security visualization tools. Snort [17] and tcpdump [18] provide industry standard network monitoring capability, but only as textual output. Ethereal also offers excellent textual output and protocol parsing, but only provides very basic visualization capability [19]. There are also a few high quality, commercial analysis tools available, which also fall into the area of network security visualization, Lancope's Stealthwatch plus Therminator system [20] is a representative example, but they proved difficult to evaluate due to the prohibitively high cost. In our research we analyzed several interesting datasets including botnet and honeynet traffic. To the best of our knowledge, there has been no work done on visualizing botnets. There has been some initial work on honeynet visualization. Honeynet monitoring is typically conducted with ethereal, snort and tcpdump, but there are two tools, primarily text-based, that provide some visualization capability. The Honeynet Security Console is a useful tool to store and analyze large amounts of data [21]. Sebek can correlate low-level host-based data (e.g. launched programs, keystrokes, accessed files) with network data to evaluate honeynet intrusions [22]. The result can be visualized as a dependency graph. Another useful, but again, primarily text based tool is OSSIM. Its primary strength lies is the correlation of multiple data streams, a feature that we do not currently provide [23].

III. System Design and Implementation

A. Design Overview

The motivation for this research stems from the need to ease the time-consuming daily monitoring of our honeynet, but at the same time, provide a tool that is useful for both general network and security analysis. Our goal is to design a visualization system that makes it possible for the analyst to see at a glance what kind of activity has occurred on a network while maintaining the ability to also provide detailed information on the activity.

The system supports two possible ways to input data. It can either display live packet capture information (real-time mode for monitoring), or it can be used to play back previously captured data (playback mode for forensics).

0-7803-9290-6/05/$20.00 ©2005 IEEE.

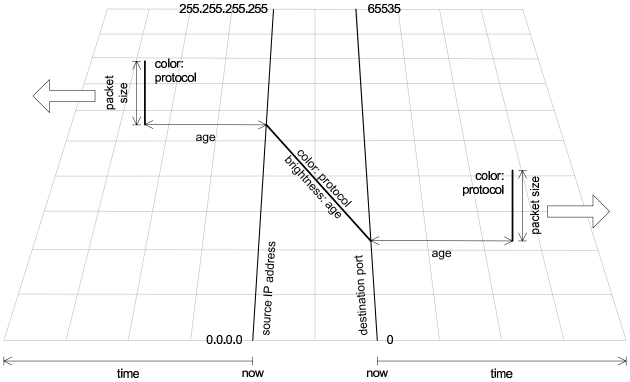


Figure 1: Visualization Overview

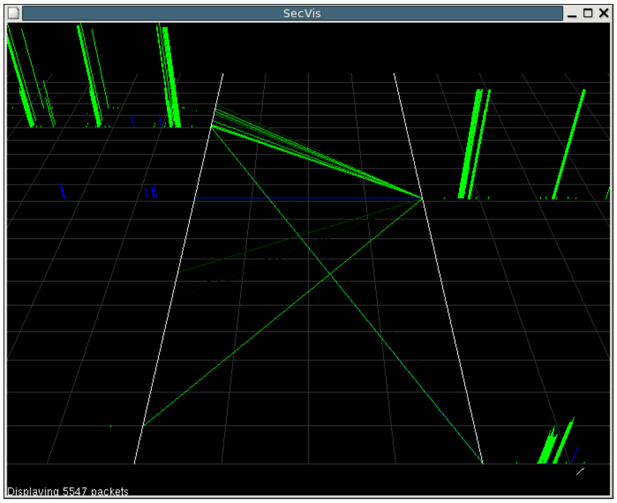
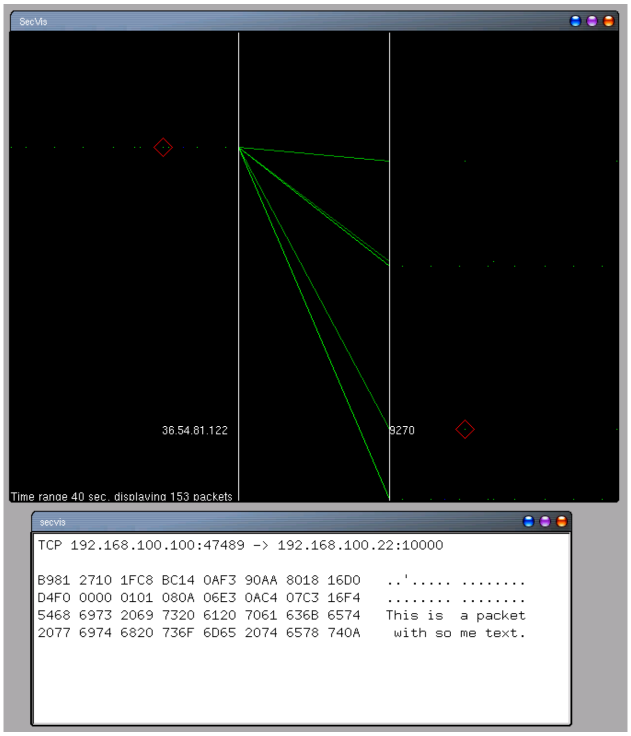


Figure 2: Example Visualization



```
TCP 192.168.100.100:47489 -> 192.168.100.22:10000

B981 2710 1FC8 BC14 0AF3 90AA 8018 16D0   ..'..... ........
D4F0 0000 0101 080A 06E3 0AC4 07C3 16F4   ........ ........
5468 6973 2069 7320 6120 7061 636B 6574   This is  a packet
2077 6974 6820 736F 6D65 2074 6578 740A    with so me text.
```

Figure 3: Detailed Packet Information

B. Visualization Design

Figure 1 shows an overview of the visualization. The bold left line (called the left base line) denotes the source IP address (0.0.0.0 - 255.255.255.255) of observed packets. We chose to implement this value as a 32-bit integer number where 0 is at the bottom and 0xffffffff is at the top. The bold right line (called the right base line) denotes the destination port number (0 - 65,535) where port 0 is at the bottom and port 0xffff at the top. Note that a port value of 0 is illegal according to the TCP and UDP specifications, but may nonetheless occur in network traffic. For each packet, a colored line links these two horizontal axes, blue for UDP and green for TCP. The line color fades to black over time so that old packets become less visible. Each packet also triggers two glyphs, represented by vertical lines in the 3D view, moving away from the left base line and the right base line over time. Each glyph represents a packet. The height of the glyph represents the packet size while the distance from the base line denotes the age of the packet. When a new packet is observed, both glyphs are on the left and right horizontal line, respectively. The glyphs move away from the base lines as the packet gets older. Figure 2 shows an example screenshot of the system. Additionally, we implemented a two-dimensional view in which packet length is not visualized. See the top application window in Figure 3.

C. Interaction Paradigm

The user can navigate with the mouse through the visualization. The mouse can be used to zoom in to a point of interest and to pan the view. In the two-dimensional view, zooming and panning only affects the y-axis so that the x-axis is still fully visible and the entire time window of observed packets can be seen by the user. In the three-dimensional view, both axes are magnified since the distortion would otherwise complicate the user's perception and navigation through the three-dimensional environment. When clicking into the window, some brief information regarding the packet closest to the mouse pointer is displayed. Both glyphs of the packet are marked with a red diamond. At the same time, protocol, source and destination IP addresses, source and destination ports, and a hex/ASCII dump of the IP payload are displayed in a separate window (Figure 3). Moreover, the IP address and port number corresponding to the current mouse pointer position are displayed next to the base lines for orientation.

In addition to the mouse, the keyboard can be used to change certain aspects of the visualization. A guiding grid can be turned on and off, which is particularly useful in the three-dimensional view mode. The time for a line to fade and for a glyph to leave the screen area can be increased and decreased. In playback mode, the speed of

0-7803-9290-6/05/$20.00 ©2005 IEEE.

the playback can be adjusted. Negative speeds allow the user to explore the dataset backwards in time.

The visualization system outlined in this paper highly profits from user interaction and animation. In addition to the screenshots provided, we plan to release a version of our program for further experimentation to the general public as a part of Georgia Tech's NETI@home project [24].

D. System Implementation

The system is implemented using three threads, a graphical user interface thread, a capture thread, and a visualization thread. Currently, the graphical user interface code only runs code to display the textual packet detail view.

The capture thread is responsible for packet capture and playback of stored packet data as well as to decode packet headers. The freely available libpcap library is used to capture packets. After decoding the packet headers, relevant information (a packet digest) is extracted and stored in a linked list. Currently, this digest includes IP source and destination addresses, transport layer protocol (TCP and UDP are supported in our current version), source and destination port, and packet length. Additionally, the first 70 bytes of the IP payload of each packet are also stored in memory so that the analyst can retrieve detailed information from each packet.

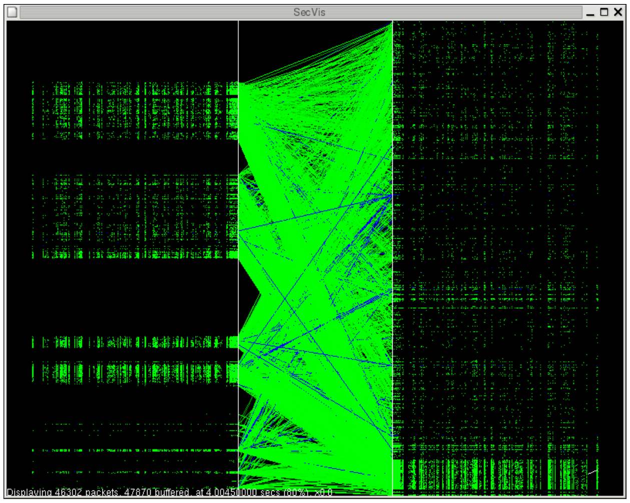
The visualization thread traverses this list, calculates the age of each packet, prunes stale packet data (real-time mode), and displays the packet information on screen.

In contrast to our tools presented in [10], the new system described here uses a retained rendering model, i.e. a digest of the packet data to be visualized is kept in memory. This design decision makes the new system computationally more expensive and less scalable on high bandwidth links, but enables features like zooming or fading packets over time.

IV. Results and Evaluation

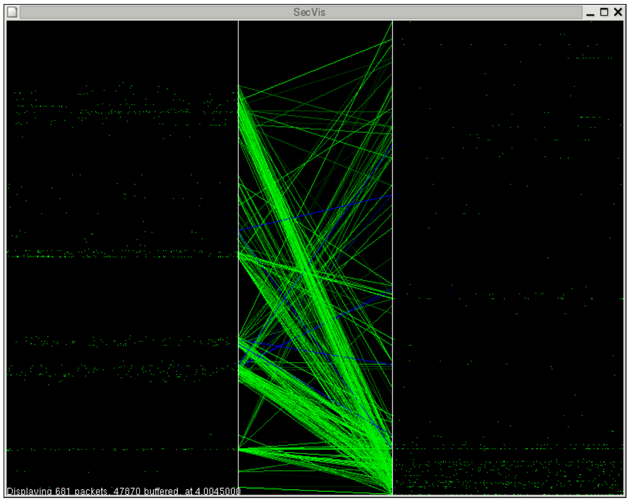
A. Backbone Traffic

In this section, we provide an overview of traffic captured on a Georgia Tech campus backbone link. In Figure 4 and Figure 5, the captured data is filtered so that only inbound data from the Internet to the Georgia Tech network is visualized. Figure 4 shows a 5 second window of captured data. The clutter on the bottom right is due to the vast amount of Microsoft Windows machines, which typically allocate ephemeral ports below 5000 for sockets they initiate. In the left and right area of the visualization, there are stripes of lower activity. We presume that these gaps are packets dropped during the capturing process due to the busy high speed link.



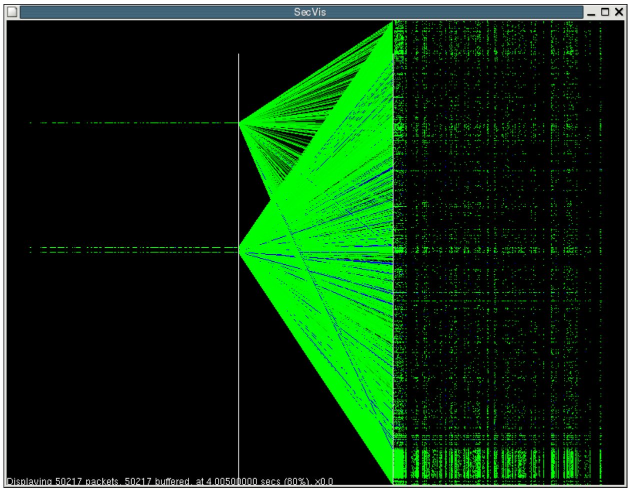
Displaying 46302 packets, 47870 buffered, at 4.00450000 secs (80%), x0.0

Figure 4: Inbound Campus Backbone Traffic (5 sec)



Displaying 681 packets, 47870 buffered, at 4.0045000

Figure 5: Inbound Campus Backbone Traffic (10 msec)



Displaying 50217 packets, 50217 buffered, at 4.00500000 secs (80%), x0.0

Figure 6: Outbound Campus Backbone Traffic (5 sec)

To reduce the clutter in the middle part of the visualization, we changed the time window visualized to 10 msec in Figure 5. It now becomes obvious that most outside machines touch low port numbers and only a few outside machines communicate to higher port numbers on the Georgia Tech campus.

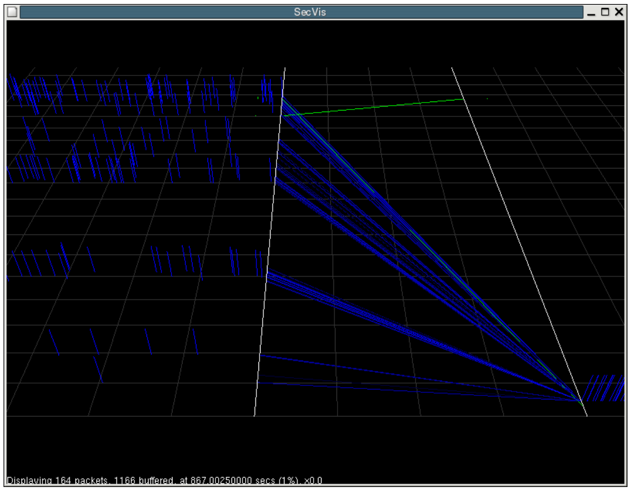
Figure 6 shows outbound traffic. The left side clearly shows traffic coming from Georgia Tech's three blocks of network addresses. The right side shows a large number of ports touched. Again, low port numbers below 5000 receive high amounts of traffic. This is due to connections initiated by off-campus machines picking ephemeral port numbers in this range. One application that results in such a pattern is peer-to-peer networking with Microsoft Windows machines. Off-campus machines connect to on-campus peers and, by default, their operating system picks low ephemeral port numbers. The result is a high volume of outbound traffic on these ports, which normally indicate client applications. Currently, as indicated by other campus network monitoring appliances, Georgia Tech's campus network carries as much peer-to-peer as Web traffic.

B. Honeynet Traffic

A honeynet is a network of honeypots, machines that are intended to be compromised, to provide the system administrator with intelligence about vulnerabilities and compromises within the network. A honeynet is placed behind a reverse firewall that captures all inbound and outbound data. The reverse firewall limits the amount of malicious traffic that can leave the honeynet. This data is surreptitiously contained, captured, and controlled. Any type of network device can be placed within the honeynet, to include configurations identical to production machines elsewhere on the network. These standard production systems are used on the honeynet in order to give an attacker the enticing look and feel of a real system. Since honeypots do not offer any legitimate services to Internet users and the Internet addresses of the honeypots are not publicly known, most traffic on the honeynet is suspicious.

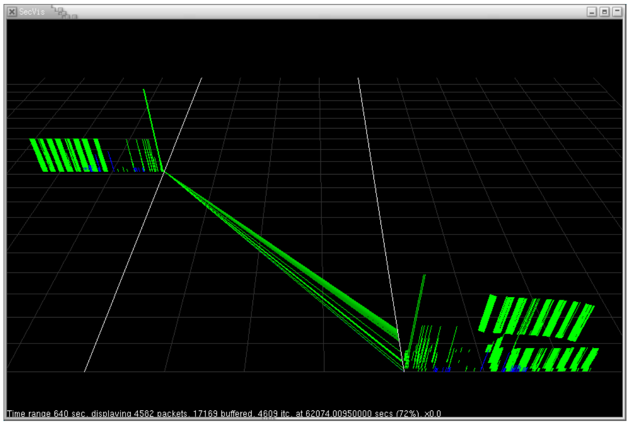
To evaluate the effectiveness of our system, we tested it with a variety of archival honeynet capture files. As an example, on January 25, 2003 the Slammer worm hit the Internet and our campus network. Slammer targeted Microsoft SQL Servers and machines running the Microsoft SQL Server Desktop Engine. The worm sent out its exploit code in 404-byte UDP packets to port 1434. Figure 7 shows 640 seconds of honeynet traffic. Clearly visible are many identically sized UDP packets (blue) targeting a single port.

On February 22, 2004, a computer on campus compromised one of the honeypots. Honeynet traffic jumped from 1.5 MB on February 21 to 4.5 MB on February 22 and back down to 0.5 MB on February 23. It took several days of studying the logs to identify exactly when the compromise occurred. Figure 8 demonstrates that using the visualization system, the compromise traffic is easily observed; the timestamp in the display directs the analyst to the exact point in the capture log to begin detailed analysis. The left IP plane shows that all traffic occurs inside the Georgia Tech address range. This significantly reduces the time required to conduct forensic analysis. A network administrator using this tool could also identify this type of abnormal traffic in real time using our visualization system.



Displaying 164 packets, 1166 buffered, at 867.00250000 secs (1%), x0.0

Figure 7: Honeynet Traffic during the Slammer Worm Attack

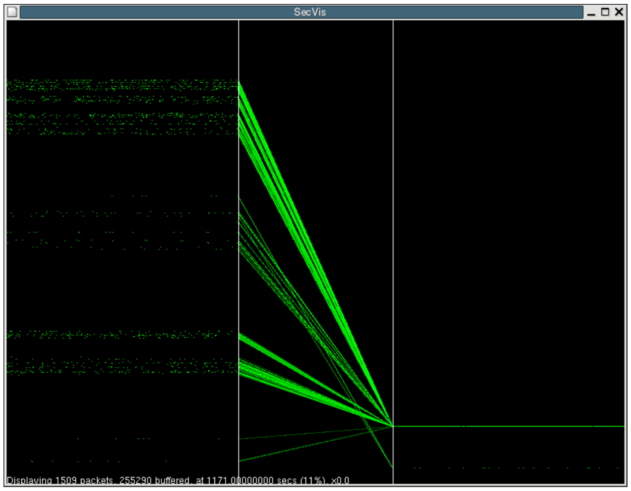


Time range 640 sec, displaying 4582 packets, 17169 buffered, 4609 itc, at 62074.00950000 secs (72%), x0.0

Figure 8: Honeynet Compromise Traffic

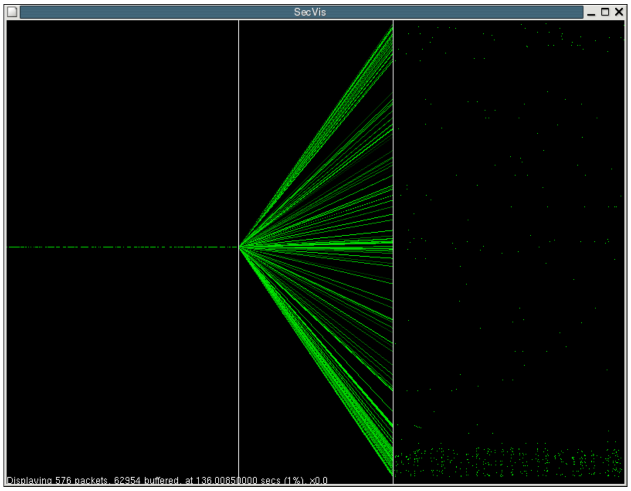
C. Botnet Traffic

Compromised machines on the Internet are often connected to so-called botnets by their attackers. The attacker maintains a large number of machines under his or her control and can leverage the connectivity of these machines to initiate distributed denial of service attacks or send out unsolicited bulk email (spam).



Displaying 1509 packets, 255290 buffered, at 1171.00000000 secs (11%), x0.0

Figure 9: Inbound Botnet Traffic

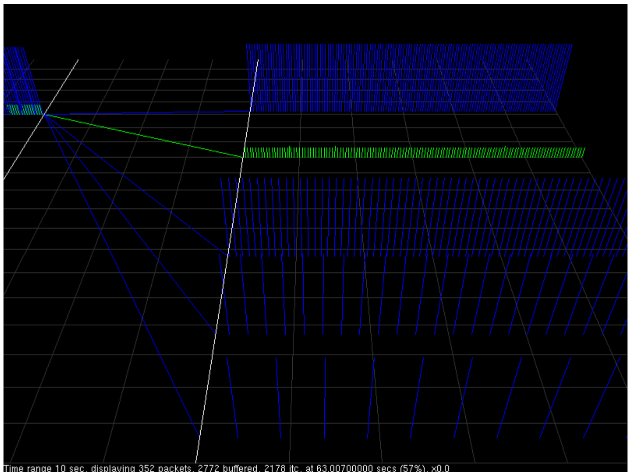


Displaying 576 packets, 62954 buffered, at 136.00850000 secs (1%), x0.0

Figure 10: Outbound Botnet Traffic

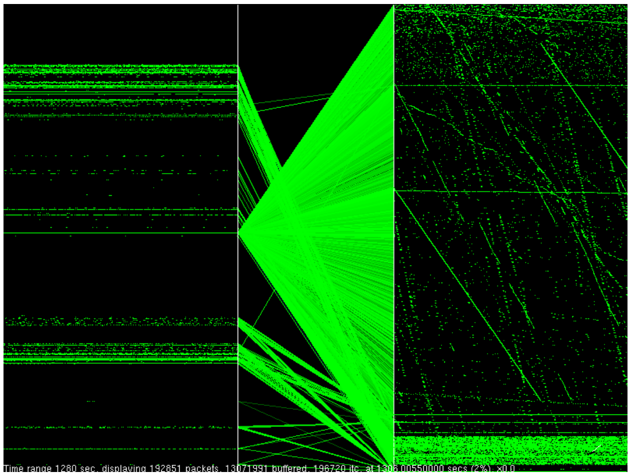
Researchers at Georgia Tech were able to gain control over several dynamic DNS entries used by botnets to contact a master server. These DNS entries have been redirected to sink machines on the Georgia Tech network. Compromised machines trying to contact their master server are directed to this local sink, and their connection attempt is logged.

Figure 9 shows the inbound traffic to one sink machine. Machines from distinctly visible netblocks on the Internet try to connect to a single port on the sink machine. This results in an image similar to the Slammer worm visualization. Similar to the honeynet, the sink machines have no configured services for public use, so communications from machines all over the Internet to a single port should raise suspicion by the analyst.



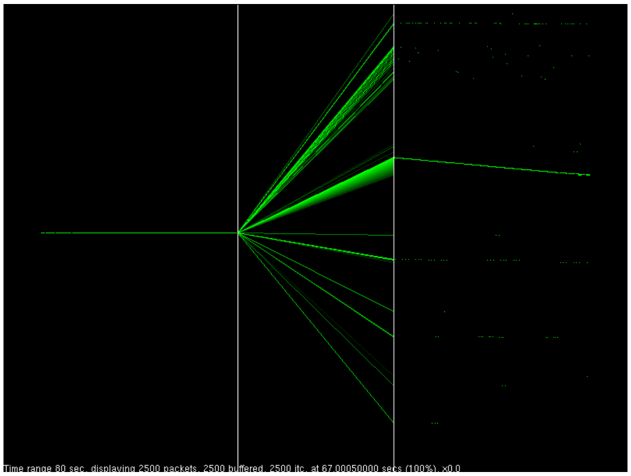
Time range 10 sec, displaying 352 packets, 2772 buffered, 2178 itc, at 63.00700000 secs (57%), x0.0

Figure 11: Constant Bitrate UDP Traffic



Time range 1280 sec, displaying 192851 packets, 13071991 buffered, 196720 itc, at 1306.00550000 secs (2%), x0.0

Figure 12: Combined Botnet/Honeynet Traffic



Time range 80 sec, displaying 2500 packets, 2500 buffered, 2500 itc, at 67.00050000 secs (100%), x0.0

Figure 13: Port Sweep

Figure 10 shows the outbound traffic from the same sink machine. A single machine, the sink, on the left side replies to numerous destination ports depicted on the right side of the visualization. Again, many machines in contact with the sink chose low ephemeral ports, which we attribute to Microsoft Windows systems being in contact with the sink. Note that this assumption is a rough heuristic only. Although we observed this phenomenon during our earlier work [10], clearly other, less popular, operating systems besides Microsoft Windows also choose such low ephemeral ports. Moreover, network

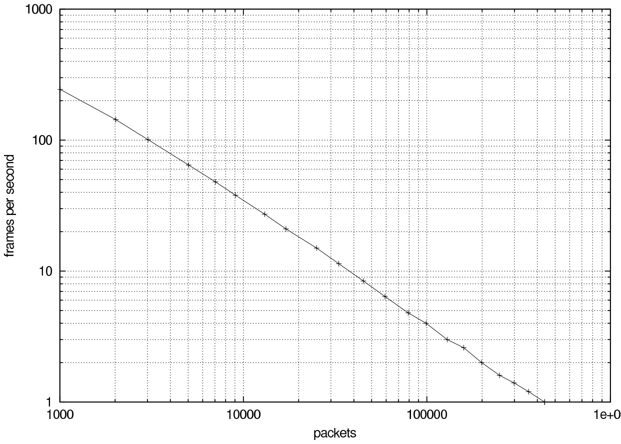


Figure 14: Frames per Second versus Number of Packets

address translation appliances, which are found in many home networks, may change the ephemeral port number chosen by the originating operating system.

D. Distinctive Visual Patterns

Our analysis has shown that, even in this straightforward visualization, distinctive patterns can easily be detected by the human observer. First, we show how flows sending at different rates can be distinguished. The visualization in Figure 11 is based on four constant bit rate UDP senders and one additional TCP sender. From bottom to top, the UDP senders send at 10,000 bps, 20,000 bps, 50,000 bps, and 100,000 bps. The TCP sender is a text-based application running over SSH. The differences in bandwidth consumption are easily visible to the human eye. The flow of packets results in a fence-like pattern loosely corresponding to the bandwidth a flow consumes.

Figure 12 shows 1280 seconds of combined botnet and honeynet traffic, both inbound and outbound. The left region of the display shows the active IP address ranges. The middle region indicates that the Georgia Tech address range (located slightly above the middle of the left base line) touches nearly all ports. Some other IP ranges touch predominantly the lower port numbers. The right region shows high activity on low port numbers, which we again attribute to the choice of ephemeral ports by Microsoft Windows. Some sloped lines indicate port sweeps. Some of these are port scans, but many are attempts of bot machines to connect to a server. The connection fails, and the bot retries. The operating system then picks a new ephemeral port number, which it generates by simply incrementing the previous one. As a result, the visualization displays a sloped line on the right region of the display. Note that increased activity on ports above 60000 is also visible. At this point we do not have a verifiable explanation for this activity.

Figure 13 shows a close up of a port sweep. The right side of the visualization indicates the sweep by a sloped line while the middle part visualized the sweep as a solid gradient triangle.

E. Scalability and System Performance

As more packets have to be visualized, the visualization gets slower. We investigate this by showing the number of rendered frames per second versus the number of visualized packets in Figure 14. Data has been gathered on a commodity 2.5 GHz Pentium 4 system with an Nvidia Quadro 2 EX based graphics card. The graph shows that the frame rate is inversely proportional to a power of the number of packets (linear dependency in the bi-logarithmic plot). We found that system interactivity is still acceptable at a frame rate of 10 frames per second, which corresponds to tens of thousands of packets. It is important to note that this limitation only constrains the number of packets that can be displayed at any given time. The datasets themselves may be far larger, up to a maximum of about 1 GByte on our prototype system. We also identified multiple possibilities to optimize the current code to further speed up the visualization.

V. Future Work

There are many areas to explore for future work. We plan to include semantic zoom capabilities. The operator will be able to retrieve very specific information the closer he or she zooms towards the visual representation of a packet. Moreover, a graphical user interface will be implemented to easily configure the mappings of multiple data sources to their visual representation. We plan to also correlate packets based on flow information to enable users to browse entire flows and analyze the content they carry as a contiguous logical unit. To remove clutter for visualization of data from high bandwidth links or large forensic files, we plan to develop efficient filter mechanisms. To integrate the system tighter into security analysis, we plan to superimpose data from other sources like IDS alerts onto the packet-level data shown.

VI. Conclusions

We presented a visualization tool that is capable of both real-time and forensic analysis of packet-level data that provides a more efficient way to browse and analyze data. The analyst can chose between different time scales and zoom into interesting areas of the data. In forensic mode, the system is capable of replaying large capture files containing days of traffic both forwards and backwards at multiple speeds. At any given time, the system provided the user a 20,000 - 100,000 packet sliding window on the dataset without compromising performance. Moreover, detailed packet information can be queried by clicking close to a glyph in the visualization.

The system has proven to be a useful tool for multiple purposes. We effectively used it in a wide variety of instances to get a rapid overview of datasets. This was a crucial aspect when analyzing our three-year honeynet capture archive of approximately 100 GBytes. Especially for these forensic analysis tasks, we found a vast decrease in the time to find points of interest in the capture files.

The system can give a much faster impression about traffic patterns and events than an analysis with Ethereal or similar programs. Once areas of interest have been identified in the data using our visualization tool, Ethereal can then be used to do a deep-protocol analysis. We believe that incorporating similar protocol analysis directly into our tool is both technically feasible and a logical direction for the future.

VII. Acknowledgements

The authors would like to thank Charles Robert Simpson, Jr. for contributing packet capture and decode code from the NETI@home project and David Dagon for providing the botnet data. We would also like to thank Lieutenant Colonel Ron Dodge and the United States Military Academy's Information Technology and Operations Center (www.itoc.usma.edu) for their continual support. This work was supported in part by the National Science Foundation Information Technology Research Award 0121643.

VIII. References

[1] Tufte, E. The Visual Display of Quantitative Information. Second Edition. Graphics Press, May 2001.

[2] Tufte, E. Visual Explanations: Images and Quantities, Evidence and Narrative. Graphics Press, February 1997.

[3] Tufte, E. Envisioning Information. Graphics Press, May 1990.

[4] Spence, R. Information Visualization. Pearson Addison Wesley, December 2000.

[5] Girardin, L. An Eye on Network Intruder-Administrator Shootouts. USENIX 1st Workshop on Intrusion Detection and Network Monitoring, 1999.

[6] Nyarko, K; Capers, T; Scott, C and Ladeji-Osias, K. Network Intrusion Visualization with NIVA, an Intrusion Detection Visual Analyzer with Haptic Integration. 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. March 24 - 25, 2002. Orlando, Florida, p. 277.

[7] The Spinning Cube of Potential Doom. http://www.nersc.gov/nusers/security/TheSpinningCube.php, last accessed March 2005.

[8] Inselberg, A. Multidimensional Detective. IEEE Symposium on Information Visualization, 1997.

[9] Marchette, D. Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint, Springer, 2001.

[10] Conti, G. and Abdullah, K. Passive Visual Fingerprinting of Network Attack Tools. ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC), pp. 45-54, Washington, D.C., October 2004.

[11] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju, VisFlowConnect: NetFlow Visualizations of Link Relationships for Security Situational Awareness. ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC), pp. 26-34, Washington, D.C., October 2004.

[12] Security Incident Fusion Tool, National Center for Advanced Secure Systems Research Group. http://www.ncassr.org/projects/sift/papers/, last accessed April 2004.

[13] EtherApe: A Graphical Network Monitor. http://etherape.sourceforge.net/, last accessed December 2004.

[14] Erbacher, R and Frincke, D. Visual Behavior Characterization for Intrusion and Misuse Detection. Proceedings of the SPIE '2001 Conference on Visual Data Exploration and Analysis VIII, CA, January 2001, pp. 210-218.

[15] Komlodi, A; Goodall, J; Lutters, W. (2004) An Information Visualization Framework for Intrusion Detection. Proceedings of the ACM Conference on Human Factors in Computing Systems (ACM CHI), ACM Press, 2004.

[16] Fink, G.; Ball, R; Jawalkar, N.; North, C. and Correa, R. Network Eye: End-to-End Computer Security Visualization. Submitted for consideration at ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec) 2004.

[17] Snort: The Open Source Intrusion Detection System, http://www.snort.org/, last accessed December 2004.

[18] TCPDump Public Repository, http://www.tcpdump.org/, last accessed December 2004.

[19] Ethereal: A Network Protocol Analyzer, http://www.ethereal.com/, last accessed December 2004.

[20] StealthWatch + Therminator. Lancope Corporation. http://www.stealthwatch.com/, last accessed April 2004.

[21] Activeworx.org, Honeynet Security Console, http://www.activeworx.org/programs/hsc/index.htm.

[22] Balas, E., Honeynet Data Analysis: A technique for correlating sebek and network data, http://www.dfrws.org/bios/day2/Balas_Honeynets_for_DF.pdf, last accessed November 2004.

[23] Open Source Security Information Management, http://www.ossim.net/, last accessed February 2005.

[24] NETI@home, http://neti.gatech.edu/, last accessed February 2005.