Using Visual Motifs to Classify Encrypted Traffic

Charles V Wright
cvwright@jhu.edu

Fabian Monrose
fabian@jhu.edu

Gerald M Masson
masson@jhu.edu

Johns Hopkins University
Information Security Institute
Baltimore, MD 21218

ABSTRACT

In an effort to make robust traffic classification more accessible to human operators, we present visualization techniques for network traffic. Our techniques are based solely on network information that remains intact after application-layer encryption, and so offer a way to visualize traffic "in the dark". Our visualizations clearly illustrate the differences between common application protocols, both in their transient (*i.e.,* time-dependent) and steady-state behavior. We show how these visualizations can be used to assist a human operator to recognize application protocols in unidentified traffic and to verify the results of an automated classifier via visual inspection. In particular, our preliminary results show that we can visually scan almost 45,000 connections in less than one hour and correctly identify known application behaviors. Moreover, using visualizations together with an automated comparison technique based on Dynamic Time Warping of the motifs, we can rapidly develop accurate recognizers for new or previously unknown applications.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*security and protection*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; I.3.8 [**Computer Graphics**]: Applications

General Terms

Security

Keywords

Traffic classification, network traffic visualization, network security

1. INTRODUCTION

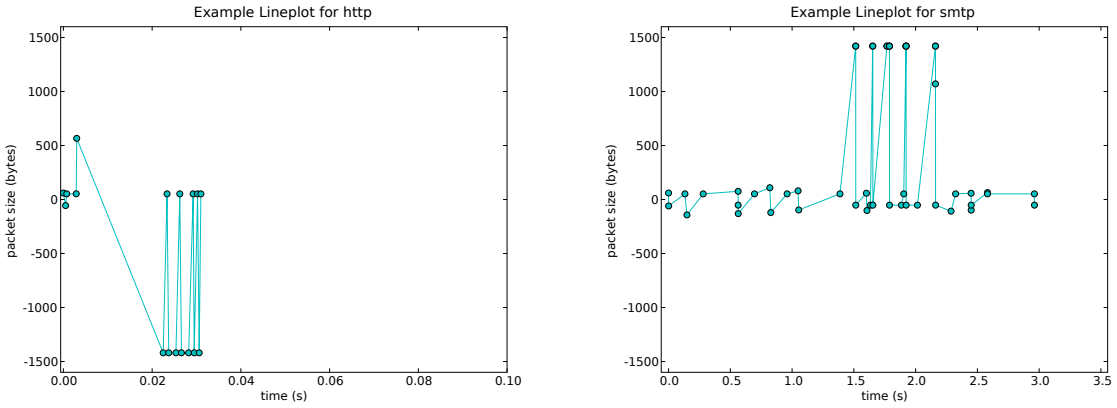
To effectively manage modern networks, administrators must be armed with the right set of tools to fully understand the nature of the activities that take place on their networks. Today, the vast amount of traffic that traverses such networks makes it impossible for a human to manually inspect every packet, and the increasing popularity of cryptography often precludes even selective inspection of suspicious traffic. Recent techniques for analysis of network traffic use machine learning to classify traffic by application [2, 10] or to find clusters of similar connections [9]. Unfortunately, the workings of these automated techniques often do not easily lend themselves to human inspection, and the statistical tools on which they are built (decision trees, Bayesian statistics, hidden Markov models, etc.) are beyond the expertise or interest of many system administrators.

While developing our own traffic classification system [15, 16], we explored new ways to visualize TCP connections so that we may better understand the nature of the traffic and be able to construct a more accurate classifier. In practice, these visualizations also provide a valuable way to verify the results of an automated classifier and a powerful tool for exploratory data analysis on anomalous or unidentified traffic. Like our previous work in automated traffic classifiers, these visualizations require very little information about each packet, and so are well suited for "inspecting" encrypted traffic. Of course, our techniques are equally applicable when the traffic remains in the clear, and has the added advantage of not needing to inspect payloads.

In this paper, we explore several methods for graphically representing on-the-wire behavior patterns of network application protocols and show how each one clearly illustrates fundamental characteristics of the given application. We demonstrate these techniques using traffic from real network traces from two large data sets. The first was was recorded in 2004 and 2005 on local area networks in the Lawrence Berkeley Laboratory (LBL) and made available in an anonymized form by Pang *et al.* [12]. The second, described by Faxon *et al.* [3], was recorded in 2003 on a wide-area link connecting the George Mason University (GMU) to the rest of the Internet.

The remainder of the paper is organized as follows. In Sections 2 and 3, we show how these fundamental behavioral characteristics, visible in the graphs as *visual motifs*, can be used to distinguish between many common application protocols. In Section 2 we focus on *transient*, or time-dependent, behavior patterns, and in Section 3 we explore several ways to visualize an application's *steady-state*, or time-independent, behavior. In Section 4, we give practical examples of how these visual techniques can be used to analyze suspicious or unidentified traffic, both to detect known behaviors and to identify new or previously unknown proto-

VizSEC'06, November 3, 2006, Alexandria, Virginia, USA.



(a) HTTP (b) SMTP

Figure 1: Line Plots for HTTP and SMTP Connections

cols and behavior patterns. We review related work in traffic classification and visualization in Section 5 and conclude in Section 6.

2. TIMELINE VISUALIZATIONS FOR TCP CONNECTIONS

We begin with techniques for visualizing single TCP connections, based on the observation that many common Internet application protocols proscribe a very specific series of client/server interactions that proceed in a linear fashion from start to finish. Upon execution of a given application protocol, these interactions are clearly visible in the sizes and timing of packets produced at the network layer and below. Moreover, fundamental differences in many of the most common application protocols make it possible to distinguish between them based on these low-level interactions.

The simplest and most familiar example of a highly linear protocol is version 1.0 of HTTP, the protocol underlying the World Wide Web. RFC 1945 defines the structure of the first version of the HTTP protocol, in which a connection typically begins with a request for a web page from the client. In return, the server replies with the contents of the requested page, and the connection is then closed. We can observe these behaviors quite clearly by looking at the sizes and arrival times of the packets generated by a very simple HTTP connection.

2.1 Visualizing Single TCP Connections

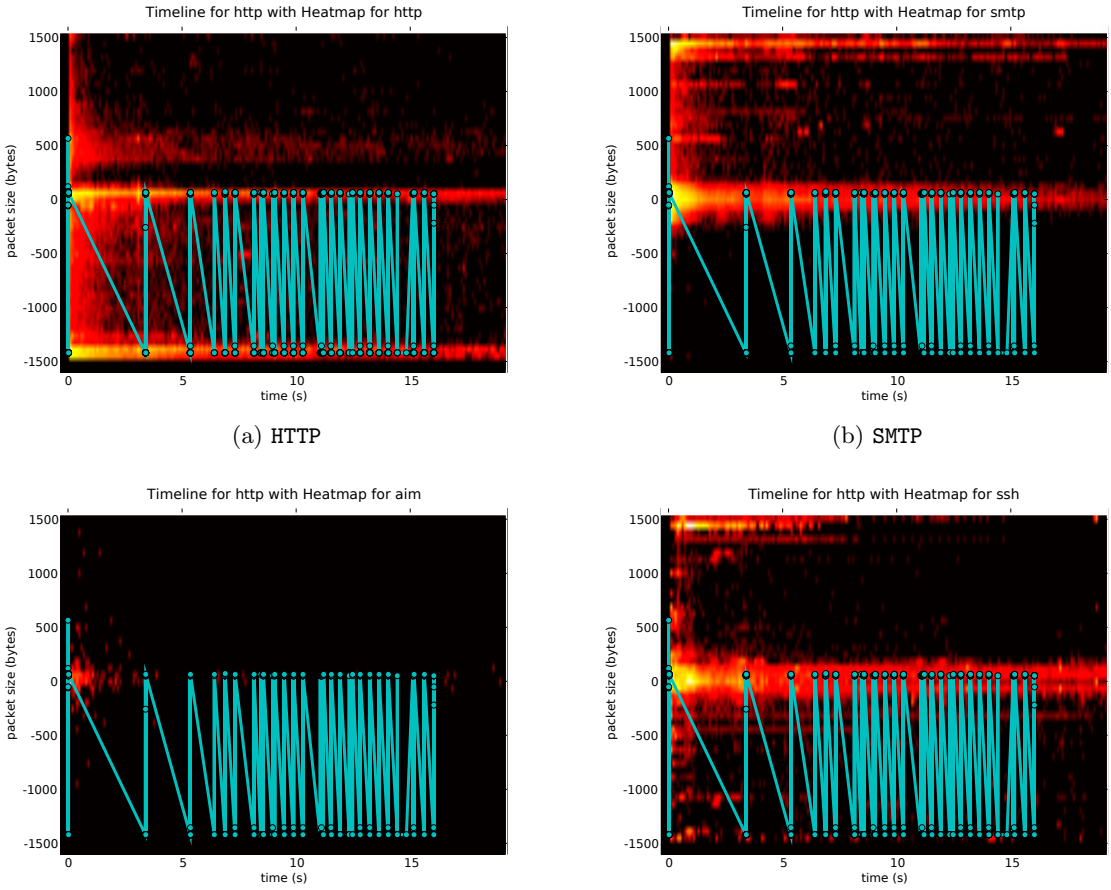
To visualize a single connection, we plot the packets in the connection on a timeline from left to right, starting with $t = 0$ at the arrival of the first packet in the connection. Each packet is represented by a point in two dimensions, where the x-coordinate corresponds to the packet's arrival time, and the y-coordinate represents the packet's size and direction. Positive Y values indicate packets from the client side of the connection; negative values of Y indicate packets from the server. In either case, the magnitude of the y-coordinate gives the packet's size in bytes, so for example a point at $(1.2, -48)$ means that a packet from the server, 48 bytes in length, arrived $1.2s$ after the start of the connection. We connect the points so that it is easy to see the progression of packets in the connection.

Despite their relative simplicity, these plots illustrate application behavior very clearly. For example, Figure 1(a) shows an HTTP connection from the LBL dataset. At the beginning of the connection, three small packets are exchanged to execute the TCP "3-way handshake." Immediately thereafter, about $5ms$ after the start of the connection, a larger packet from the client arrives carrying the HTTP request, and the server's reply arrives starting at $21ms$.

Figure 1(b) shows a SMTP connection from the same dataset. This connection looks very different from that in Figure 1(a) because the SMTP protocol behaves very differently from HTTP. The primary difference is that, while HTTP is a "pull" protocol, SMTP is a "push" protocol, whereby clients contact servers to offer data unrequested. Another difference is that SMTP requires more cross-talk between client and server to set up connection parameters and to identify the sender and recipient of the mail before the message itself is transferred. Both of these properties are again visible in the line plot (Figure 1(b)). As before, three small packets execute the 3-way handshake at the beginning of the connection, then between $0.1s$ and $1.1s$, a series of slightly larger packets are exchanged before the start of the bulk data transfer—this time from client to server—at $1.4s$.

2.2 Viewing Multiple Connections Simultaneously

A straightforward extension to the above technique for visualizing a single connection can be used to succinctly display the overall pattern of behaviors exhibited in a set of many connections. One naïve approach would be to simply overlay the line plots for all the connections in a single plot, but due to the noisy nature of Internet traffic, any discernible patterns would be lost in the resulting graphs. An alternative is to simply omit the lines from the plots and instead draw only a set of points to indicate where the packets arrived. Doing so gives us essentially a scatter plot of points in two dimensions. Unfortunately, in scatter plots, it is often difficult to distinguish between areas of moderate density (where many packets have been plotted very near to each other) and areas of very high density (where many packets have been plotted on top of each other). Therefore, to increase the utility of our graphs, we use color to indi-



(a) HTTP (b) SMTP

(c) AIM (d) SSH

Figure 2: Line plot for HTTP Connection with Heatmaps for HTTP, SMTP, AIM, and SSH

cate, at each pixel, the number of packets which have been plotted at the given x,y-coordinates. Starting with a dark background, we increase the brightness of a pixel each time a packet maps to it. The end result is a heatmap plot, showing "hot" areas of very high packet density in bright colors and "cold" areas of low density in dark colors.

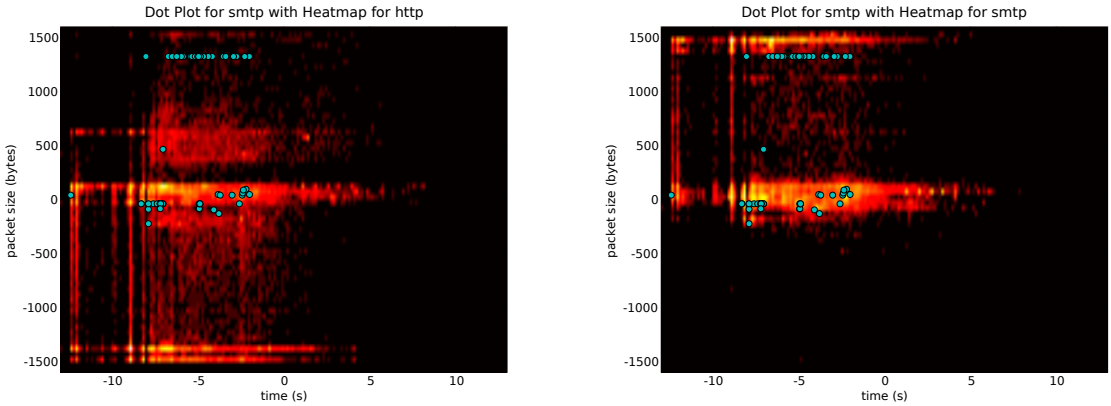
To compare a new TCP connection to the known behavior of a set of protocols, we simply overlay a line plot for the connection on top of the heatmap for each protocol. The best-matching protocol should have bright spots in its heatmap where the connection has packets, and there should be few packets in dark areas of the heatmap.

Figure 2 shows an HTTP connection compared to heatmaps for HTTP, SMTP, the AOL Instant Messenger (AIM), and SSH, all generated from traces in the LBL dataset. In each of the heatmaps, small TCP ACK packets from both sides of the connection make the area around the x-axis very bright. AIM connections in the LBL data tend to be quite short and carry only a few small packets, so the bright red region only extends for about one second after connection initiation. This region is brighter in the heatmap for SSH (Fig. 2(d)) than in those for HTTP (Fig. 2(a)) and SMTP (Fig. 2(b)) because SSH's most common behavior is to send many small packets back and forth between the client and server, sending and echoing keystrokes.

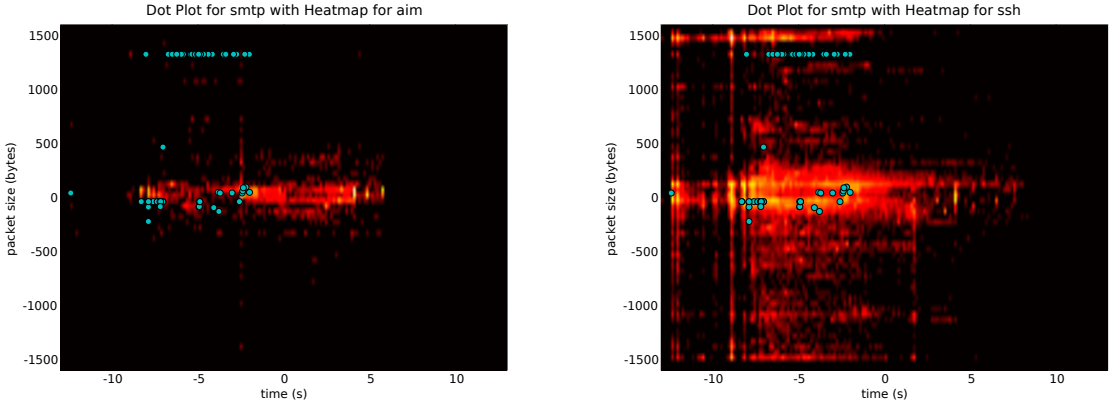
In the SMTP graph, this highlighted region around the x-axis is more diffuse, including packets up to about 100 bytes in both directions, because in SMTP, the two sides exchange several messages ("MAIL FROM ...", "RCPT TO ...", etc.) before engaging in bulk data transfer of the email message itself. In the heatmap for SMTP, the bright line at the top of the graph indicates that data flows mainly from client to server. Notable features in the HTTP heatmap are the burst of color corresponding to the client request, brightest at (0.2, 600), and the bright line across the bottom that illustrates the bulk transfer of the web page from the server. The red area below the x-axis is caused by the random distribution of the size of the last data packet in the page's bulk transfer.

3. IDENTIFYING NON-LINEAR BEHAVIORS

The aforementioned timeline graphs are motivated by the idea that most application protocols describe a linear progression of messages from the start of a connection to its finish. However, many protocols, particularly those designed for human interaction, allow for much more free-form behaviors which may be highly *non*-linear. Telnet and interactive SSH, for example, send most of their packets as a direct result of some user action. Specifically, upon a keypress, the



(a) HTTP (b) SMTP



(c) AIM (d) SSH

Figure 3: Heatmaps for Packet Unigram Frequencies with Dot Plot for an SMTP Connection

client machine sends a packet to the server, which then responds by sending a packet back to the client to echo the character to the user's screen. The server then waits for the arrival of the next packet from the client, and the process repeats.

To identify this and other interactive behaviors, it may be more appropriate to instead view these protocols in terms of their steady-state behaviors, rather than as time-dependent processes. To this end, we have also developed techniques for visualizing the unigram and bigram frequencies observed in the packets generated by a given protocol.

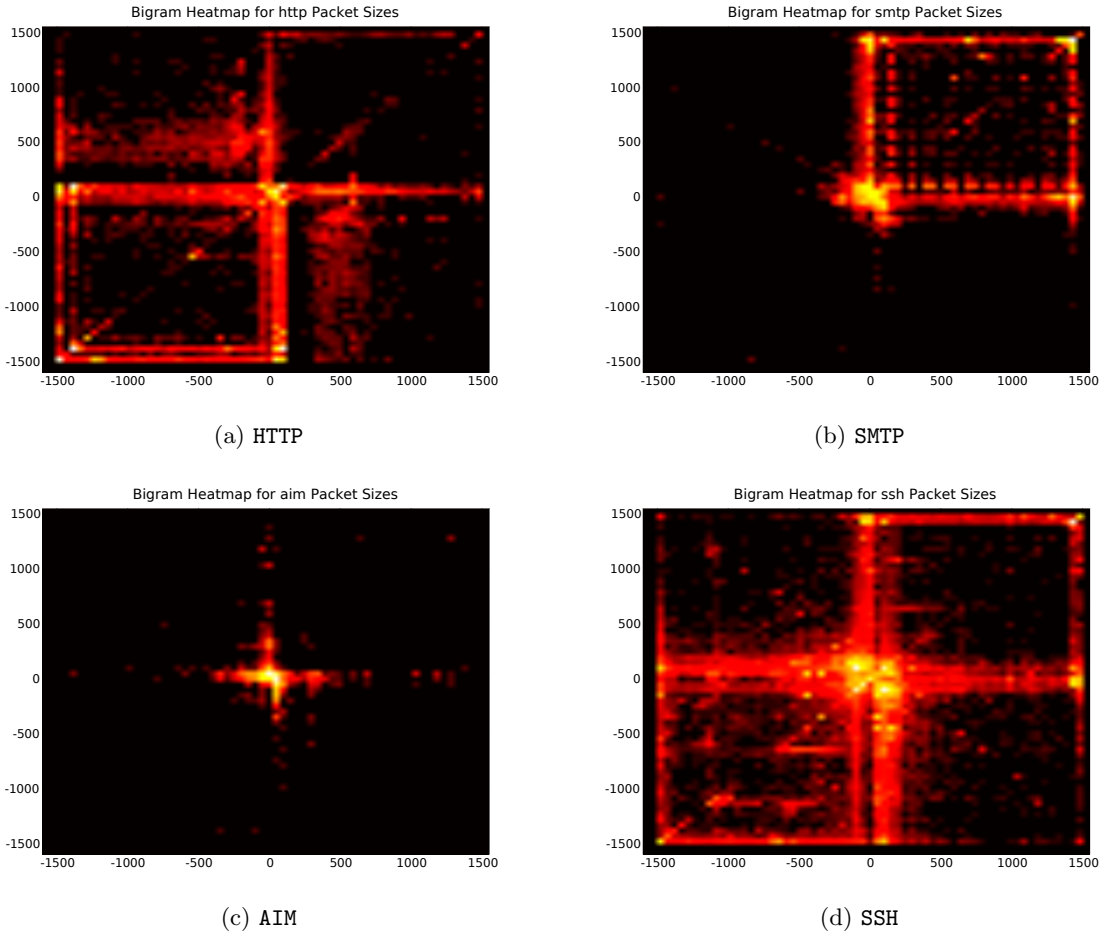
3.1 Visualizing Unigram Packet Frequencies

The simplest way to measure the steady-state behavior of an application protocol is in terms of its unigram packet frequencies—that is, to see what kinds of packets it tends to generate, on average, assuming that the packets are each generated independently and all follow the same distribution.

To visualize the unigram packet frequencies exhibited by a given network protocol, we use heatmaps similar to those in Section 2, with time on the x-axis and packet size on the y-axis. However, instead of the time elapsed since the start of the connection, we set each packet's x-coordinate as the log of its *interarrival time*, that is, the time elapsed since the arrival of the previous packet in the connection. Because interarrival times are bounded on the left (at zero) but unbounded on the right, and often follow a heavy-tailed distribution, it is convenient to take the log of the interarrival times, rather than work with them directly. As before, we start with a dark background, and brighten pixels when packets map to them, resulting in bright areas where many packets occur, and dark areas where there are few or no packets.

To compare a new TCP connection to the profiles of a known protocol, we simply plot the packets from the new connection as colored dots on top of the protocol's heatmap. To determine which protocol best matches the observed TCP connection, we look to see which heatmap is brightest in the areas where the new connection has colored dots.

Figure 3 shows the dot plot for an SMTP connection overlaid on top of heatmaps for AIM, HTTP, SMTP, and SSH. Again, properties of both the HTTP request (the burst of 200-300 byte packets from the client side with short delay) and reply (the bright horizontal line of big packets from the server with small interarrival time) are visible in Fig 3(a). In Fig 3(b), SMTP has a broader region around the x-axis for ACKs and mail negotiation, as well as several large packets with short interarrival times in the top and left sections, indicating the bulk data transfer from client to server. As one



(a) HTTP (b) SMTP

(c) AIM (d) SSH

Figure 4: Bigram Heatmaps for Packet Sizes

might expect of an interactive application, AIM (Fig 3(c)) tends to generate mostly small packets with longer interarrival times. SSH, in Fig 3(d), shows features of both interactive use, with small packets from both directions and a wide variety of interarrival times, and of bulk data transfers in both directions.

Clearly, in this case, the heatmaps for SMTP and SSH match the newly-observed SMTP connection much better than HTTP or AIM. SMTP seems a better match than SSH because the SMTP-heatmap is slightly brighter near some of the packets, particularly those in the cluster in the lower left. Also, the dots for this connection are all in the middle and upper regions of the graph, where almost all SMTP packets fall, whereas SSH seems equally likely to produce packets in the lower half of the graph.

3.2 Visualizing Packet Bigrams

The unigram heatmaps in the previous section are based on the assumption that packets for each protocol are independent and identically distributed. But because TCP uses positive acknowledgments to implement reliable data transfer, and because many application protocols have a general query/response structure, both the sizes and the interarrival times of consecutive packets are often highly correlated. For example, in a bulk data transfer, large data packets are usually followed either by other large data packets with very small delay, or by small TCP ACK packets with delay on the order of one round trip time.

To visualize these causal relationships between consecutive packets, we can look at the *bigram frequencies* observed in the packets generated by each protocol. That is, we look to see what kinds of packets the protocol tends to generate, given the properties of the previous packet in the connection. For this, we can once again use heatmaps, but in a two dimensional plot, we are limited to visualizing only one feature of the traffic at a time. We can generate heatmaps for the consecutive packet sizes observed in the traffic, or for consecutive interarrival times, but we cannot visualize both features at the same time in the same plot.

Figure 4 shows bigram frequencies for packet sizes. In each plot, the first packet's size is shown on the x-axis, and the second packet's is on the y-axis. The protocols each show very distinctive patterns. HTTP (Fig. 4(a)) tends to follow client ACK's with data packets from the server, visible in the thick vertical stripe in the lower half of the graph. The HTTP requests are visible in Fig. 4(a) as a blur in the lower right quadrant centered around X=500 bytes. Again we can see in Fig. 4(b) how SMTP tends to send most of its large packets from the client to the server, and that these are commonly followed by other large client-to-server pack-

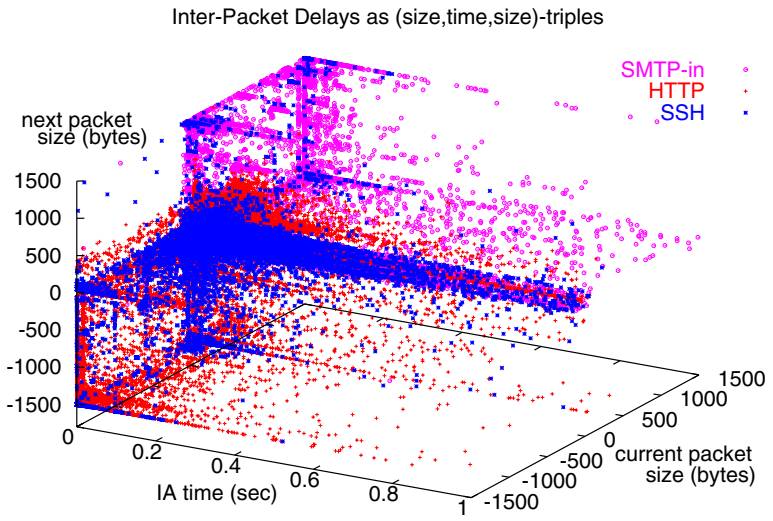


Figure 5: 3D Plot of Packet Inter-arrivals for `HTTP`, `SMTP`, and `SSH`

ets, or by small ACK packets from the server. Fig. 4(c) shows how `AIM` traffic is dominated by small packets, and that any slightly larger packets are typically preceded and followed by smaller ones. `SSH` shows a much more diffuse behavior pattern. Small packets are very common in both directions (the orange regions around both axes), and these often follow other small packets, as evidenced by the bright yellow clusters around the origin. `SSH` also shows evidence of back-to-back large packets flowing in both directions, as well, in the lines of color around the outside edges of the upper right and lower left quadrants; this behavior is likely due to `SCP` or other tunneled bulk data transfer protocols such as `rsync` or `CVS`.

Unfortunately, the heatmaps in Figure 4 do not contain any information about the timing of the packets generated by the given protocols. Packet timing has proven to be extremely useful in gleaning information from network traffic: for distinguishing between interactive and machine-driven connections and for identifying related pairs of connections [20]; for inferring `SSH` keystrokes to crack login passwords [14]; and even for deriving the private keys of `SSL`-enabled web servers [1]. We also believe that timing is an important feature to consider for traffic classification.

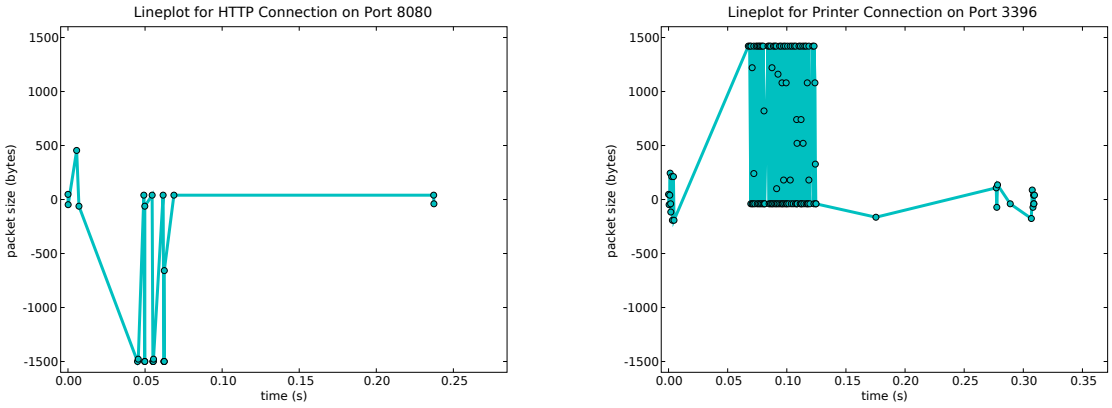
We could create heatmaps similar to Figure 4 for bigrams of interarrival times, but we find timing information alone to be too noisy to be very useful. Instead, by moving to 3-dimensional plots, we can convey more concisely what a typical pair of consecutive packets looks like for a given protocol. We represent each bigram of packets as a point in 3-dimensional space, with coordinates defined by the first packet's size (X), the interarrival time between the two packets (Y), and the second packet's size (Z). (Indeed, the heat maps in Figure 4 can be seen as a type of projection of this three-dimensional data onto the X-Z plane.) Figure 5 gives an example of such a plot for three very differently-behaving protocols: `HTTP`, `SMTP`, and `SSH`.

4. APPLICATIONS

We now highlight several practical uses for the visual traffic classification techniques developed in this paper. In practice, such techniques might be used by network administrators as an anomaly detection tool, to verify, for example, that all traffic to their web servers on TCP port 80 does indeed exhibit the characteristic behavior patterns of `HTTP`; a connection which looks more like `Telnet` or `SSH` may be evidence of a break-in. The visual techniques could also be used to perform misuse detection, to find unauthorized servers or to identify traffic generated by prohibited applications such as chat programs or video games. Applications to anomaly detection are relatively straightforward; therefore we focus here on applications for misuse detection. Specifically, we show how the new visual methods can enable a human operator to recognize instances of known protocols on nonstandard ports and to identify new or unknown protocols and behaviors.

As a first application of these techniques, we use visualizations to identify the protocols used in traffic observed on nonstandard ports in the LBL traces [12]. Because the equipment used to record a packet trace may sometimes drop or omit packets, we do not require that each connection start with a full TCP 3-way handshake. Similarly, we do not omit duplicates or re-order out-of-order packets based on TCP sequence numbers. Rather, we simply map packets into "connections" based on a 5-tuple of: transport protocol, source IP address, source port number, destination IP address, and destination port number. This approach has the advantage of requiring low overhead and is in line with our goal of using as little per-packet data as possible, in order to be robust against various kinds of encrypted connections.

From a single day (Jan 6, 2005) in the LBL data [12], we extracted over 4400 such "connections" with both source and destination port numbers greater than 1024. Because high-numbered ports are often used for prohibited services such as chat programs, instant messengers, peer-to-peer file



(a) Port 8080

(b) Port 3396

Figure 6: Line plots for Possible `HTTP` Connections

sharing, rogue mail relays or unauthorized web servers, these connections give us a good way to test the visual techniques for detecting the types of traffic that an administrator would be interested in finding.

For each of these connections, we generate a line plot and dot plots and compare them to the typical behavior patterns of known application protocols. One potential pitfall associated with using line plots with these data collection methods is that if the first packet we see in a connection is not really the first packet in the connection and was not sent by the host which initiated the connection, then we may confuse the client and the server for each other. Therefore, when looking at the line plots, we must keep in mind that a connection *may* be drawn with the two directions inverted, so that the image appears reflected about the x-axis.

Visually, `HTTP` has one of the most easily recognizable behavior patterns. The round trip time between the medium-sized request from the client side and the first large packet carrying the start of the requested page from the opposite side of the connection creates a very distinctive diagonal line across the x-axis; this is followed by several steeper lines connecting data packets from the server and ACKs from the client.

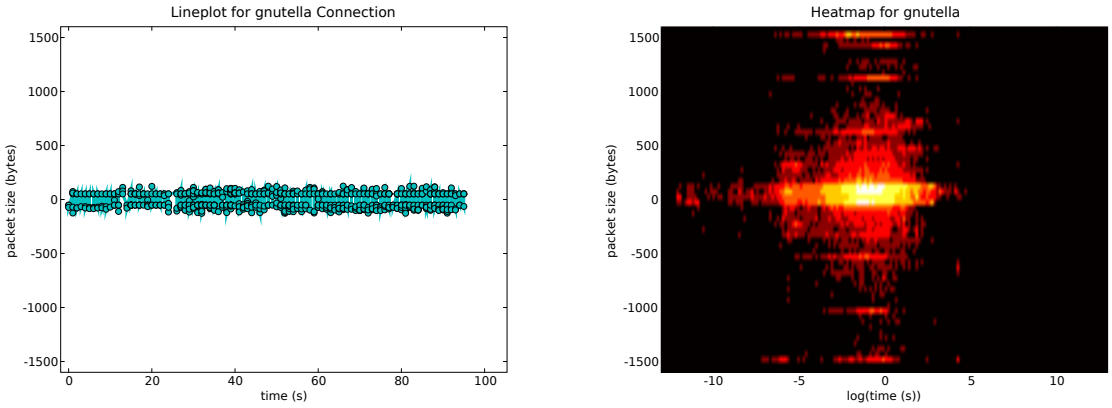
By scanning the line plots looking for `HTTP`'s distinctive visual motif, in about 15 minutes we conservatively identified 39 connections out of the 4400 as potential instances of `HTTP`, many of which may have been drawn "inverted" due to a dropped first packet. Figure 6 shows two examples of likely `HTTP` connections. Without packet payload data, we cannot determine with certainty which of these connections carry `HTTP` traffic. However, based on the port numbers they use, 15 of them, all on TCP port 8080 at the anonymized IP address 128.3.148.116, probably are in fact `HTTP`. Figure 6(a) shows one such connection. Sixteen others, on ports 3396 and 9100, which we initially suspected might have been "inverted" `HTTP` connections, appear to be connections to network printers. For an example of these printer connections, see Figure 6(b). Four connections on port 6200 may be instances of the Oracle Notification Service, and four other connections remain unidentified, but look much more similar to the suspected Oracle and printing connections than to known web traffic.

The most distinguishing behavioral characteristic for separating `HTTP` connections from the others is the size of the packet early in the connection which precipitates a data transfer in the opposite direction. In web traffic, this packet carries the `HTTP` request, and is typically 500-600 bytes in length; for the traffic which we suspect belongs to other protocols, this packet is usually much smaller, typically under 100 bytes. For an example of this difference, see Figure 6. In Fig 6(a), the "request" packet before the long diagonal line is almost 500 bytes in size; the corresponding packet in Fig 6(b) is only about 150 bytes.

In a similar experiment, we generated line plots of all TCP connections not on well-known ports during one 10-minute midday trace from the GMU dataset [3]. In this short period, we found over 39,000 such connections. The majority of this traffic seems to be the result of port scanning or other failed connections to ports that are not listening; most are likely not full connections and do not appear to carry real application interactions. In these, we see only a few small packets at long, random intervals, represented in the line plots as long, near-horizontal lines near the x-axis.

However, among the noise, we quickly discovered a distinctive, repeated behavior pattern unlike that of any of the classical application protocols in our initial set [15]. Most (though not all) of the connections which exhibit this pattern occur on TCP port 6346, which is often used by the Gnutella peer-to-peer file-sharing protocol. Inspection of the first few bytes of payload in the first packets of these connections indicates that this traffic is, in fact, version 0.6 of the Gnutella protocol. Figure 7(a) shows an example of such a Gnutella connection. As seen here, a connection between Gnutella nodes tends to generate many small, closely spaced packets traveling in both directions, likely carrying search queries and requests to join the P2P network.

Our initial visual inspection of the line plots identified 22 Gnutella connections. We selected one of them as a template, and used Dynamic Time Warping [13] to compare the other unknown connections to it. By visually examining the 100 best-matching connections, as ranked by DTW score, we found more than 30 other instances of Gnutella which visually look very similar to the template. With over 50 examples of the Gnutella protocol, we have sufficient training



(a) Example Line Plot

(b) Unigram Heatmap

Figure 7: Profiling Gnutella

data to now apply our earlier techniques [15] using profile hidden Markov models to construct an automated detector for the protocol. We can also generate visual profiles of Gnutella traffic, in the form of timeline, unigram (Fig. 7(b)) and bigram heatmaps, and 3D bigram plots (Fig 8). In this fashion, the visualization techniques can be used together with machine learning approaches to quickly develop recognition ability for newly-discovered protocols and behavior patterns.

To further test our new techniques, we used the same template Gnutella connection to check for other instances of the protocol in the "enterprise traces" collected at Lawrence Berkeley Lab [12]. Because LBL is a US government facility, we can reasonably expect that peer-to-peer file-sharing applications such as Gnutella would be prohibited by policy and thus extremely rare (if present at all) on the network. GMU, on the other hand, is a residential university, with hundreds of student machines in dorms, so we expect that the Gnutella traffic would be relatively common there.

Again, we use DTW to compare new connections to the Gnutella template. To determine the degree of similarity required for a "match", we generated line plots for all the unknown connections from the original day in the GMU dataset and sorted them in order of decreasing similarity to the template connection. Then, by visual examination, we set the detection threshold equal to the score of the connection immediately following the last similar-looking connection. We stress that this somewhat *ad-hoc* method for setting detection thresholds, should not be used as a replacement for rigorous techniques founded in Statistics and Information Theory, but rather as an aid to human operators for use in situations when statistical results are inconclusive or not available.

In over 4,400 connections from the LBL dataset, the automated Dynamic Time Warping recognizer flagged fewer than 30 as matches with the template Gnutella connection. Visual inspection of the line plots for these "candidate" connections indicates that only about half of them look sufficiently similar to warrant further investigation, and none of them are as similar to the template as the connections identified at GMU. This example illustrates the utility of the visual techniques, not only for the initial detection and profiling of a new behavior or protocol, but also for double-checking the results of an automated classifier.

5. RELATED WORK

The problem of traffic classification has received much attention in recent years. One of the first approaches, presented by Zhang and Paxson in 2000, uses cleartext packet payloads to identify application protocols [19]. Moore and Papagiannaki [10] present a slightly more sophisticated approach which also relies on payloads for its classifications. Moore and Zuev [21, 11] provide a more robust traffic classification scheme based on Bayesian analysis techniques, which requires access only to packet header data. A decision tree approach by Early *et al.* [2] also uses only header information.

Our own traffic classification methods [15, 16] use profile hidden Markov models and require even less data, including only packet size, direction and arrival time. These lean data requirements enable traffic classification on traffic encrypted with an end-to-end cryptographic protocol such as SSL or TLS, or in the words of Karagiannis *et al.* [5], "in the dark". In [5], they take a different view of traffic classification and focus on identifying the role of *hosts* in the network by recognizing patterns in the IP addresses and ports with which each host communicates. The visual traffic classification techniques in this paper, like any of the above approaches, could be used in concert with the system in [5] to build a comprehensive traffic classification system which considers both high-level behaviors of the hosts and low-level behaviors of the individual packet flows or connections.

The graphlets in [5] are also possibly the only published method for effective visualization of the inner workings of a traffic classification scheme, although various forms of visualization have been used for similar traffic analysis tasks. The graphlets themselves are quite similar to the parallel axis view used by Yin *et al.* [17] for visualization of NetFlow data. Another approach for visualizing NetFlow data is given by Lakkaraju *et al.* [7].

McGregor *et al.* [9] present techniques for visualizing single TCP connections and for clusters of many connections. In particular, they use scatter plots of packet size versus interarrival time for individual connections; our unigram

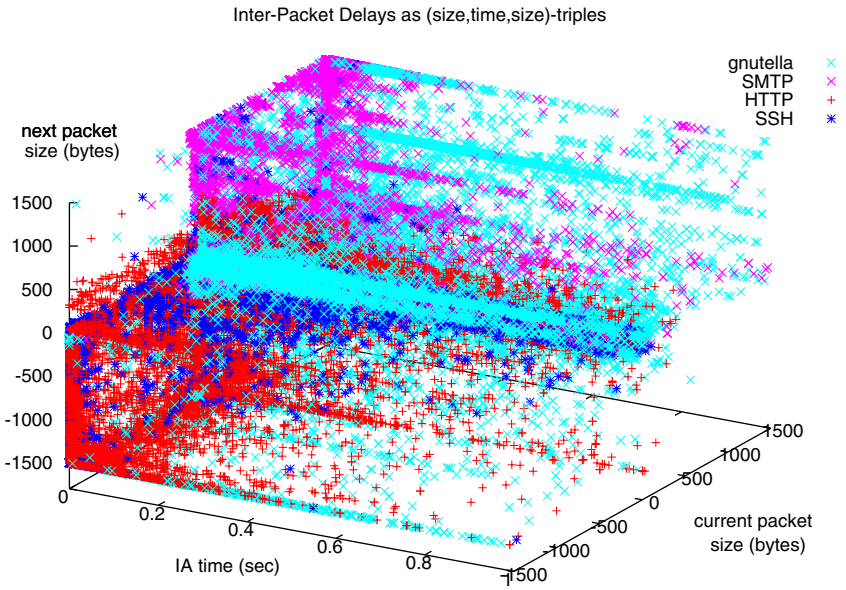


Figure 8: 3D Plot for Gnutella with `HTTP`, `SMTP`, and `SSH`

heatmaps in Section 3.1 can be seen as a logical extension of this work to enable the visualization of the overall trends in massive datasets. The preliminary ideas put forth by Goldring [4] on using scatter plots for visualizing network traffic are also related. While the author of that work was unsure of how such graphs could be used or indeed what they might mean, we have assigned meaningful semantics and developed a practical application for these techniques in network security monitoring.

Yoda and Etoh [18] use line plots of packet arrival time versus TCP sequence numbers to identify pairs of connections which attackers use to relay interactive `SSH` traffic along of a chain of "stepping stone" hosts to disguise their location and the true source of their attacks. This technique is similar to our line plots in Section 2, although the graphs in [18] only track one side of the connection at a time and rely on TCP sequence numbers instead of packet sizes. More distantly related is work by Lin *et al.* [8] on visualizing and finding motifs in time series databases.

6. CONCLUSIONS AND FUTURE WORK

We introduce new ways to visualize and identify characteristic behavior patterns in encrypted network traffic. Using real traffic traces, we show that these characteristic behaviors, visible as *visual motifs* in the graphs, can be used to distinguish between application protocols, to find known protocols or behaviors in unlabeled traffic, and to discover new protocols and rapidly develop the ability to recognize them in other traffic. Work is currently underway to integrate these tools into an interactive GUI application.

Our current techniques are robust against end-to-end, session layer cryptographic protocols such as SSL or TLS, where encryption occurs above the network layer. As future work, we plan to investigate ways to effectively visualize and classify traffic which has been encrypted at the network layer or below, as is the case with `SSH` tunnels or IPsec ESP [6].

7. ACKNOWLEDGMENTS

We graciously thank the Statistics Group at George Mason University for providing access to their packet traces, and Pang *et.al.* for making their anonymized logs available. This work is supported by NSF grant CNS-0546350.

8. REFERENCES

[1] D. Brumley and D. Boneh. Remote timing attacks are practical. In *Proceedings of the* 12^{th} *Usenix Security Symposium*, pages 1–14, August 2003.

[2] J. Early, C. Brodley, and C. Rosenberg. Behavioral authentication of server flows. In *Proceedings of the* 19^{th} *Annual Computer Security Applications Conference*, pages 46–55, December 2003.

[3] D. Faxon, R. D. King, J. T. Rigsby, S. Bernard, and E. J. Wegman. Data cleansing and preparation at the gates: A data-streaming perspective. In *2004 Proceedings of the American Statistical Association*, August 2004.

[4] T. Goldring. Scatter (and other) plots for visualizing user profiling data and network traffic. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 119–123, New York, NY, USA, 2004. ACM Press.

[5] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *ACM SIGCOMM*, August 2005.

[6] S. Kent and R. Atkinson. RFC 2406: IP encapsulating security payload (ESP), November 1998.

[7] K. Lakkaraju, W. Yurcik, and A. J. Lee. NVisionIP: netflow visualizations of system state for security situational awareness. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 65–72, 2004.

[8] J. Lin, E. Keogh, and S. Lonard. Visualizing and discovering non-trivial patterns in large time series databases. *Information Visualization Journal*, 4(2):61–82, April 2005.

[9] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *The 5^{th} Anuual Passive and Active Measurement Workshop (PAM 2004)*, April 2004.

[10] A. Moore and K. Papagiannaki. Towards the accurate identification of network applications. In *The 6^{th} Anuual Passive and Active Measurement Workshop (PAM 2005)*, March 2005.

[11] A. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. In *Proceedings of the ACM SIGMETRICS*, June 2005.

[12] R. Pang, M. Allman, V. Paxson, and J. Lee. The Devil and Packet Trace Anonymization. In *ACM Computer Communication Review, 36(1)*, pages 29–38, January 2006.

[13] L. Rabiner, A. Rosenberg, and S. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(6):575–582, December 1978.

[14] D. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and SSH timing attacks. In *Proceedings of the 10^{th} USENIX Security Symposium*, August 2001.

[15] C. Wright, F. Monrose, and G. M. Masson. HMM profiles for network traffic classification (extended abstract). In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 9–15, October 2004.

[16] C. V. Wright, F. Monrose, and G. M. Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, Special Topic on Machine Learning for Computer Security. (to appear).

[17] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju. Visflowconnect: netflow visualizations of link relationships for security situational awareness. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 26–34, 2004.

[18] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. In 6^{th} *European Symposium on Research in Computer Security (ESORICS)*, pages 191–205, October 2000.

[19] Y. Zhang and V. Paxson. Detecting back doors. In *Proceedings of the 9^{th} USENIX Security Symposium*, pages 157–170, August 2000.

[20] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9^{th} USENIX Security Symposium*, pages 171–184, August 2000.

[21] D. Zuev and A. Moore. Traffic classification using a statistical approach. In *Proceedings of the Passive and Active Measurement Workshop (PAM2005)*, March/April 2005.