Traffic Classification Using Visual Motifs: An Empirical Evaluation

Wilson Lian
Dept. of Computer Science
University of North Carolina at Chapel Hill
wwlian@gmail.com

Fabian Monrose
Dept. of Computer Science
University of North Carolina at Chapel Hill
fabian@cs.unc.edu

John McHugh
Dept. of Computer Science
University of North Carolina at Chapel Hill
mchugh@cs.unc.edu

ABSTRACT

In this paper, we explore the effectiveness of using graphical methods for isolating the differences between common application protocols—both in their transient and steady-state behavior. Specifically, we take advantage of the observation that many Internet application protocols proscribe a very specific series of client/server interactions that are clearly visible in the sizes and timing of packets produced at the network layer and below. We show how so-called "visual motifs" built on these features can be used to assist a human operator to recognize application protocols in unidentified traffic. From a practical point of view, visual traffic classification can be used, for example, for anomaly detection to verify that all traffic to a web server on TCP port 80 does indeed exhibit the characteristic behavior patterns of `HTTP`, or for misuse detection to find unauthorized servers or to identify traffic generated by prohibited applications. We present our technique for building a classifier based on the notion of visual motifs, and report on our experience using this technique to automatically classify on-the-wire behavioral patterns from network flow data collected from a campus network. Specifically, we analyze over 1 billion flows corresponding to over 5 million sessions on nearly 200 distinct ports, and show that our approach achieves high recall and precision.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces; I.5.4 [**Pattern Recognition**]: Applications; K.6.5 [**Security and Protection**]: Network monitoring

General Terms

Traffic visualization, traffic analysis, evaluation

Keywords

Information visualization, security, network management

1. INTRODUCTION

Over the last few decades, packet timing and size have been used in intriguing ways to glean useful information about the contents of network communications. In fact, with the increased use of cryptographic techniques on the Internet, so too have we have witnessed a marked increase in techniques for unveiling interesting facets of encrypted communications [6, 18] using only those features that remain intact after encryption. Some notable examples include detecting what web pages are being transferred over `ssl`-encrypted connections [14, 9], inferring the keystrokes being entered in an `ssh`-session as a user types her password [13], and more recently, inferring the language of—or even the presence of particular phrases in—encrypted VoIP communications [17, 21], to name a few.

To better assist security practitioners in diagnosing aberrant behavior on their networks, informative (and interactive) visualizations have been widely sought after. Indeed, several venues (e.g., "FloCon" and "VizSec") have sprung up to provide forums for operational analysts, software developers, and researchers to explore next-generation ideas for analyzing large volumes of network traffic. Spurred by the need for practical solutions to real-world challenges, countless approaches (e.g., [19, 3, 8, 10, 20, 15, 2, 5, 11]) have been suggested in the last few years for providing new and insightful ways for visualizing such traffic.

In what follows, we investigate a particular strategy for classifying encrypted network traffic for the purpose of detecting the presence of application protocols running outside of their normal port ranges. Obviously, the ability to reliably detect instances of various application protocols—without inspecting the contents of packets as they fly across the network—would be of tremendous practical value to network administrators, traffic engineers, and the networking community at large. This information can inform network administrators' decisions to investigate and remedy possible misbehaving clients or to deploy appropriate countermeasures.

The approach we explore focuses on application protocol visualization and naturally extends the ideas first proposed by Wright *et al.* [19]. That work proposed the use of packet size, direction and order as features, which were shown to be useful features when building HMM-based classifiers [18] for single flow detectors. Like Wright *et al.*, our work is guided by a handful of assumptions regarding the nature of the traffic being classified. Specifically, we assume that the application protocols that we are interested in classifying use the Transmission Control Protocol (TCP) at the transport layer. Additionally, we assume that any encryption in use takes place at the transport layer or higher and, for efficiency, employs a stream cipher which preserves the relative sizes of payloads. Even with this limited set of information, we show that protocol classification using *visualizations* built on these features can be quite successful, assuming, of course, that the adversary is not making

VizSec '10, September 14, 2010, Ottawa, Ontario, Canada.

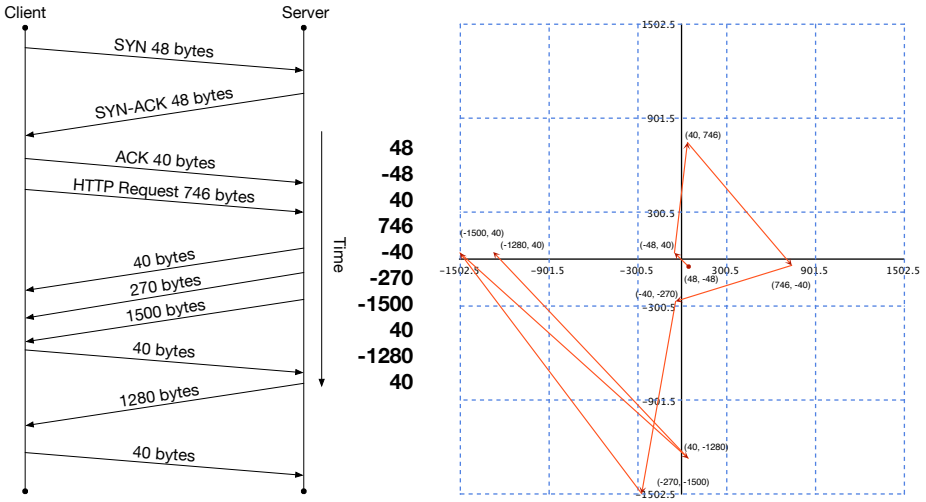


Figure 1: Example binning for a fragment of a typical HTTP session. Arrow heads indicate the order in which the data points are added. Cartesian axes are shown as solid lines, and the boundaries of 601×601 heatmap bins are shown as dashed lines.

deliberate active attempts to change a protocol's behavior to more closely resemble something else [1, 16]. In what follows, we provide the first extensive evaluation using visual motifs as features for classifying a large collection of real-world traffic, and highlight several practical considerations (e.g., on parameter selection) that were not considered before.

The remainder of the paper is structured as follows. We discuss our traffic visualization and modeling technique in § 2. We outline our methodology for traffic classification "in the dark" [6] in § 3 and evaluate its efficacy in § 4. We discuss limitations in § 5. In § 6, we conclude and propose future extensions.

2. OVERVIEW

In what follows, we summarize the key problem addressed by our work and the approach we take in addressing the problem at hand. Loosely speaking, we are interested in examining a collection of flows for an arbitrary TCP port in order to determine if these flows may belong to an application of interest to us. The application protocols we consider may include any number of common protocols, for example, HTTP, SSH, SMTP, Kazaa, AIM, etc. For the purposes of this paper, we are generally interested in inspecting traffic from a particular host destined for a given TCP port (e.g., connections to port 22 over the course of a day) to determine if the client is genuinely running the well-known application for that port (e.g., SSH) or if its behavior conforms more to that of a prohibited application (e.g., BitTorrent) configured to accept connections on the port in question.

Visual Motifs

The approach we take in this paper is to build a traffic classifier where the key features are the visual motifs first suggested by Wright *et al.* [19]. Loosely speaking, these motifs exploit the structured nature of application protocols to create vibrant representations of application protocol behavior.

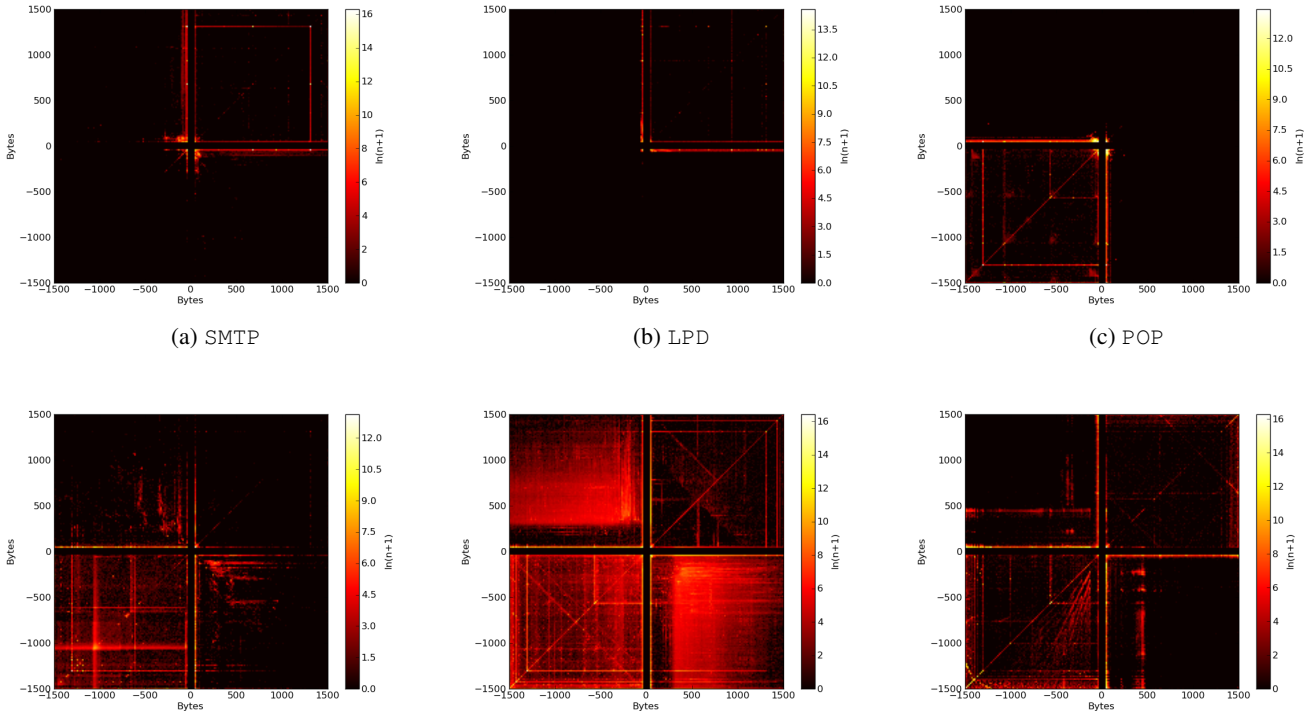
For pedagogical reasons, we briefly revisit the idea of visual motifs. Visual motifs provide a representation of aggregated network traffic from multiple sessions and connections using so-called "heatmaps" that are constructed as follows: Starting with a black background, ordered pairs are mapped to locations on the heatmap, which are progressively brightened. In that way, those regions that have a high concentration of ordered pairs eventually stand out from the black background. We refer to the locations on the heatmap as "bins".

Network traffic is encoded as a heatmap by simply treating the heatmap as a Cartesian coordinate plane and using the size and direction of consecutive IP packet pairs as coordinates. The magnitude of each coordinate is determined by the size of the packet, and the sign is defined as positive for packets from the client to the server and negative for packets in the reverse direction (i.e., from the server to the client).

Each ordered pair falls into exactly one bin. We ensure this property by making the bins rectangular and uniform in size, with a fixed height and width that must be an odd positive integer. Furthermore, we adopt the convention that the x and y axes must pass through the centers of the bins that lie along them; bins cannot overlap.

Figure 1 illustrates how 9 ordered pairs are derived from 10 packets in a typical HTTP session. Note that the first packet size in each pair becomes the x-coordinate, and the second packet size becomes the y-coordinate. These ordered pairs are mapped onto a Cartesian plane with 601×601 heatmap bins, whose boundaries are denoted by dashed lines. Notice that no bin boundaries fall on integer values. Since the boundaries of the bins on the edges of the figure extend beyond 1500 bytes—the maximum length of an IP datagram that can be sent over Ethernet [4]—we choose not to plot the heatmaps beyond the range $[-1500, 1500]$ in the x and y directions.

Figure 2 shows heatmaps for aggregated network traffic on six distinct TCP ports, labeled by the application protocol most commonly associated with that port. In Figures 2a and 2b, we observe that in the heatmaps for protocols dominated by traffic from client to server—such as the Simple Mail Transfer Protocol (SMTP) and the Line Printer Daemon (LPD) protocol—the top right quadrant is the brightest. Figures 2c and 2d show the reverse for predominantly server to client protocols such as the Post Office Protocol (POP) and the Real Time Streaming Protocol (RTSP). Protocols such as the HyperText Transfer Protocol (HTTP) and the FastTrack protocol employed by the peer-to-peer file sharing application Kazaa are inherently more bi-directional, resulting in heatmaps that are not



(a) SMTP (b) LPD (c) POP

(d) RTSP (e) HTTP (f) Kazaa

Figure 2: Heatmaps for 4 common application protocols.

dominated by any single quadrant (Figures 2e and 2f).

Modeling Protocol Behavior

We employ the traffic structure captured by the heatmaps to model application protocols using only the information that remains intact after encryption. As noted earlier, Wright *et al.* [19] observed that message exchanges for distinct instances of a given protocol share a common structure, and these structures differ between disparate application protocols. We use these structural differences as the basis for our classification scheme.

To better understand how this works, consider, for example, a "push" protocol such as SMTP whose traffic contains message text and embedded objects and consists mainly of packets from the client to the server. Typically, these packets sizes correspond to the maximum transmission unit (MTU), followed by small TCP ACK packets from the server. That contrasts sharply with that of an interactive SSH connection, in which packets in both directions are small, containing key strokes and their corresponding acknowledgments from the server. Key to the preceding discussion is the fact that our ability to observe these message structures relies only on packet size, direction, and ordering; the availability of payload data neither aids nor harms our efforts.

To model the behavior of an application protocol, we consider the distribution of the data points used to create a stable heatmap for that protocol. Next, we describe our method for measuring the difference between an arbitrary pair of such distributions. However, before doing so we first introduce some notation.

Given a protocol model $\mathcal{M}$ with n bins, we number each bin from $\{1, \ldots, n\}$ and denote the number of data points that map to bin i as $\mathcal{M}_{\phi_i}$. Furthermore, we define $\mathcal{M}_\tau$ as the number of data points used to construct the entire model $\mathcal{M}$. Given two protocol models $\mathcal{A}$ and $\mathcal{B}$ with the same bin layout, we measure the degree by which they differ using the L_1 distance between the equivalent relative frequency histograms as:

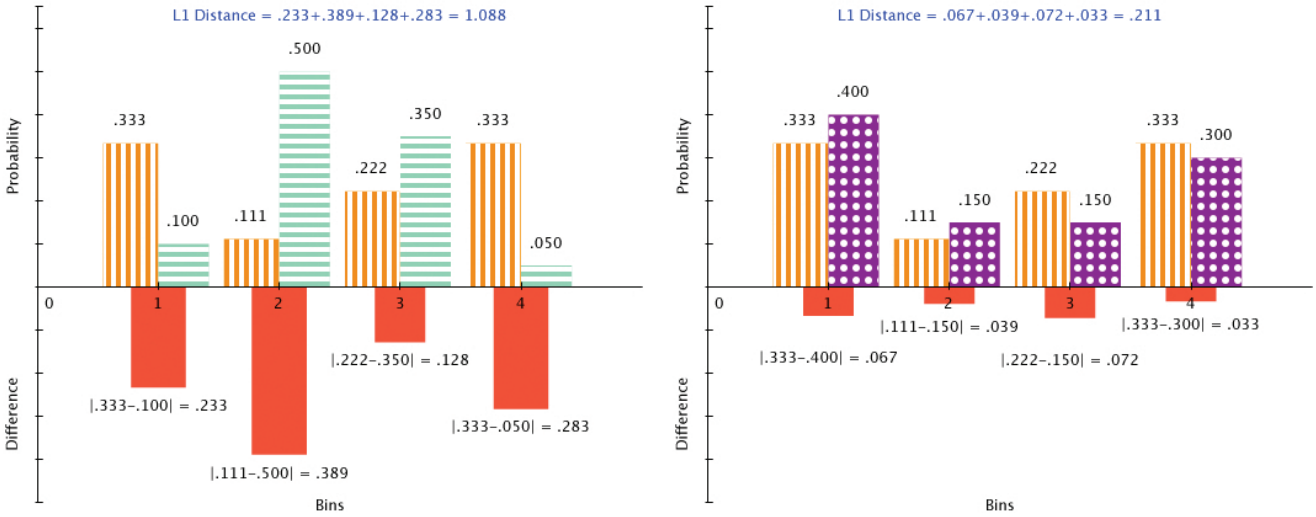
$$\sum_{i=1}^{n} \left| \frac{\mathcal{A}_{\phi_i}}{\mathcal{A}_\tau} - \frac{\mathcal{B}_{\phi_i}}{\mathcal{B}_\tau} \right|$$

This metric is symmetric (i.e., $L_1(\mathcal{A}, \mathcal{B}) = L_1(\mathcal{B}, \mathcal{A})$) and is bounded by the range $[0, 2]$. If $L_1(\mathcal{A}, \mathcal{B}) = 0$, then the models have the same distribution of data points, and their heatmaps appear identical (even though they may have been constructed using different volumes of data). If $L_1(\mathcal{A}, \mathcal{B}) = 2$, then there are no bins for which more than one protocol model contains data points. In other words, the bins in the model could be partitioned into those to which only data points from $\mathcal{A}$ fall, those to which only data points from $\mathcal{B}$ fall, and those to which no data from either $\mathcal{A}$ or $\mathcal{B}$ falls.

3. METHODOLOGY

Once potentially-aberrant traffic has been identified using the visualization techniques discussed in §2, we can use a properly-trained classifier to ascertain the nature of the traffic. In this section, we describe how our protocol modeling scheme and model comparison metric are employed in a traffic classifier capable of detecting the presence of an application protocol in network traffic.

To train the classifier to detect the presence of an application protocol P_i which operates on some well-known TCP ports (say, $p_{i_1}, \ldots, p_{i_k}$), we randomly select a large number of TCP sessions destined for ports $p_{i_1}, \ldots, p_{i_k}$, construct ordered pairs from these



(a) (b)

Figure 3: Example classification of a testing model (vertical stripes) against (a) a model from a different protocol (horizontal stripes), and (b) a model from the same protocol (white spots).

sessions, and bin the ordered pairs from all of the sessions into a single heatmap model (see Figure 1). Our evaluation of the exact number of sessions that qualifies as a "large number of TCP sessions" will be revisited in § 4.1.3.

To classify the traffic, we create ordered pairs from randomly-selected sessions from the traffic and bin the ordered pairs into a single testing heatmap model (see Figure 1) until some threshold quantity, λ, of ordered pairs have been aggregated. We compute the L_1 distance between the testing data's heatmap model and each of the training models, $\mathcal{M}_1, \ldots, \mathcal{M}_m$, resulting in a sequence of distances $d_1, \ldots, d_m$, and their standard deviation, σ. Let d' and d'' be the smallest and second smallest distances, respectively. The training model belonging to d' is considered a match to the testing sample if $|d' - d''| \geq k\sigma$ holds, where k is a non-negative constant confidence threshold. If d' does not lead d'' by the confidence threshold, then the classifier is not able to make a decision and simply returns $\perp$.

For example, Figure 3 illustrates the classification of a testing model (shown with vertical stripes in both Figures 3a and 3b) against two training models—Model 1 with horizontal stripes and Model 2 with white spots. The height of each bar in a model indicates the probability that a random ordered pair used in the construction of the model falls into the heatmap bin represented by that bar (i.e., the relative frequency of ordered pairs falling into each bin). The solid bars below the x-axis indicate the contribution to the L_1 distance for each bin in the histogram. Observe that Model 1 is clearly a poorer match to the testing model than Model 2. Indeed, in this illustrative example, it is the case that the testing model and Model 2 are merely instances of the same protocol.

4. EVALUATION

To evaluate the accuracy of our classifier, we perform empirical evaluations using two large datasets. The first dataset (denoted `dartmouth`) was used to explore appropriate parameter values for training set sizes, confidence thresholds, and heatmap bin sizes. As shown later, the results from the analysis using that dataset are then used to set parameters for our evaluation from a disparate set of labeled traffic (denoted `darpa`). Before proceeding, we elaborate on the specifics of each dataset.

The `dartmouth` dataset contains real Internet traffic data gathered in January and February 2004 from wireless sensors located on the campus of Dartmouth University. The traces used provide IP header information in addition to TCP flags and timing information which were used to reconstruct TCP sessions. Unfortunately, these traces do not provide payload data, and so we assume that the application protocols transmitting on a particular port correspond to the application protocols that use that well-known or registered port (i.e, the traffic on port 22 is for `SSH`). We used 15 days of weekday data for the three consecutive weeks from January 19–February 6, 2004. Overall, the network trace comprises 707 GB of traffic observed across 64,214 ports. There were a total of 193 ports that exhibited at least one million TCP packets, constituting a total of roughly 1.2 billion packets. We were able to reconstruct 5.2 million TCP sessions from traffic on these ports, and selected the ten ports with the most sessions—port 25 (`smtp`), 80 (`http`), 88 (`kerberos`), 110 (`pop`), 135 (`dce`), 139 (`netBIOS`), 443 (`https`), 445 (`mds`), 902 (`vmware`), 1214 (`kazaa`) for analysis.

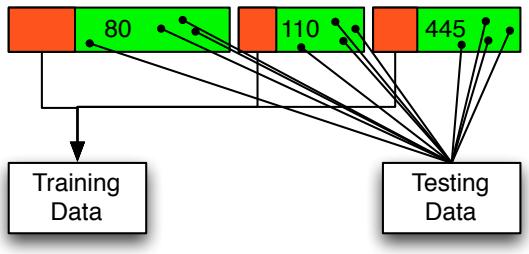
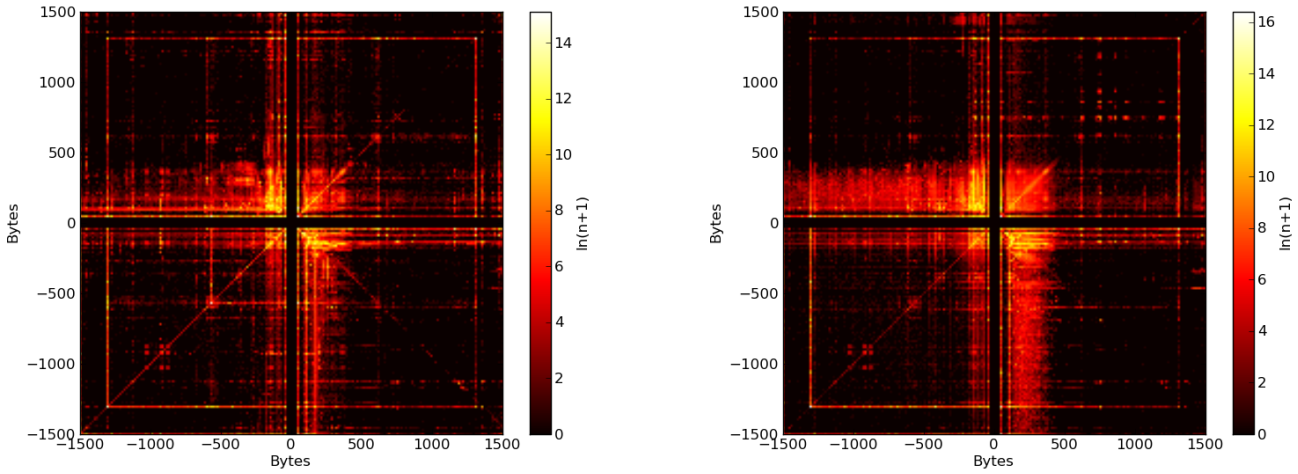


Figure 4: The classifier is trained on a randomly-selected percentage of the available sessions for each protocol. Testing models are constructed using randomly-selected sessions until λ data points are collected.



(a) `NetBIOS Session Service` (b) `Microsoft Active Directory Service`

Figure 5: Heatmaps for traffic on ports 139 and 445 in the `dartmouth` dataset

Our second dataset is from the *1999 DARPA Intrusion Detection Evaluation* conducted at the Lincoln Laboratory at MIT [1]. Specifically, we use weeks, 1, 3, 4 and 5. The data contains full network headers and payloads, and comprises roughly 12 GB of network traffic in 60 million packets observed across 10,274 ports. We reconstructed TCP sessions for all ports exhibiting traffic in this dataset, resulting in 1,580,416 sessions. Since many ports had only a handful of sessions, we selected only traffic to those ports that had enough traffic to perform meaningful analyses; resulting in the selection of 6 ports, namely, 21 (`ftp`), 23 (`telnet`), 25 (`SMTP`), 79 (`finger`), 80 (`http`), 110 (`pop3`).

The analysis that follow consists of cross-validation tests among 50 independent trials. In each trial, we trained the classifier using a percentage of the available sessions (sampled at random) for each protocol, then randomly sample another set of sessions for testing (see Figure 4). To build the testing model, we sample sessions until we have $\lambda = 45,000$ data points[2]. We then use the training models to classify the data, and compute the number of true positives (TP), false positives (FP), and false negatives (FN) with respect to each trained protocol. These results are used to compute the precision (i.e., $\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) across the 50 trials. Finally, we report the average precision and recall values across all trained protocols as the overall precision and recall for the set of trials.

4.1 Parameter Selection

As the observant reader would have noticed, our evaluation requires several parameters to be set before we can evaluate the effectiveness of the classifier. In what follows, we elaborate on how we selected values for these parameters.

[1]While the use of this dataset has come under some fire [12] as being unrealistic for intrusion detection purposes, we simply use the label data for each benign protocol as a way to gauge the promise of our techniques

[2]The choice of λ was derived empirically — at that value, we observe $\approx 5\%$ increase in accuracy over using as few as 10,000 points. Beyond that the improvement is marginal. See Appendix A for a table showing of the number of sessions required to build a testing sample with various values of $\mathcal{M}_\tau$

4.1.1 *On Setting the Bin Size*

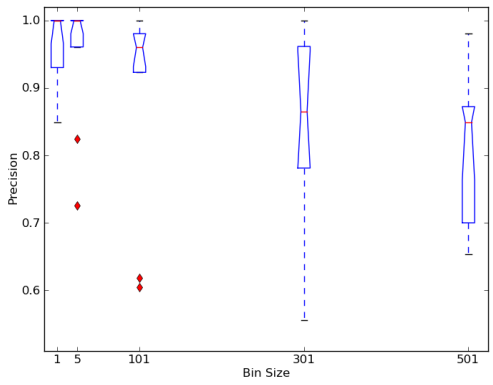
Adjusting the size of the bins that comprise the heatmap models has several implications. For one, it affects the resolution at which the classifier can distinguish between sequences that share much of the same packet ordering and direction, but vary primarily in the sizes of packets. Intuitively, one would surmise that small heatmap bins (e.g., 1×1, 3×3, etc.) would provide a more detailed representation of the data being analyzed. While true, the high resolution afforded by small heatmap bins comes at a price. Specifically, the total number of bins in the heatmap varies inversely with the product of the bins' height and width. Hence, the amount of time required to calculate the L_1 distance between a given pair of heatmap models increases with decreasing bin size.

This means that any implementation of our traffic classifier (say, on dedicated hardware) would merit careful selection of the bin sizes to avoid requiring a prohibitively-expensive quantity of memory without compromising the classifier's ability to accurately and reliably detect protocols. Figure 6 provides box plots of the precision and recall across the 10 ports in the `dartmouth` dataset as we vary the bin sizes. Notice that the smaller sizes perform well, but since a heatmap with 1×1 bins takes 25 times longer to calculate an L_1 distance than a heatmap with 5×5 bins, we sacrifice a small amount of precision and recall for speed.

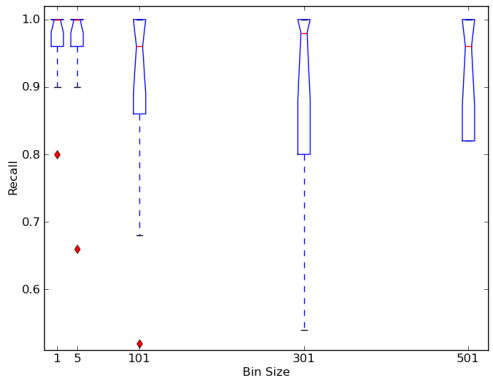
4.1.2 *On Selecting the Confidence Threshold*

The confidence threshold k allows one to specify by how much the first choice training candidate must be a better fit to the testing model than the second choice candidate. Recall that the confidence threshold k is a non-negative constant that is scaled by the standard deviation σ of the distribution of L_1 distances between each of the training models and the testing model.

We scale k by σ so that the confidence threshold can expand and contract with the distribution of the L_1 distances. The intuition is that if the distribution of L_1 distances exhibits a small variance, we do not want to require the first choice candidate to lead by a large constant amount. Alternately, if the L_1 distances are spread out, we want the first choice candidate to lead by more before making a decision. Scaling by σ allows us to normalize the amount of

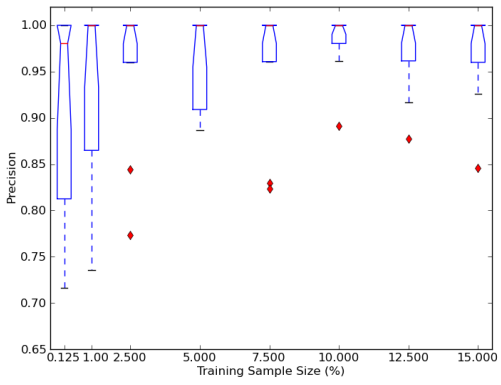


(a)

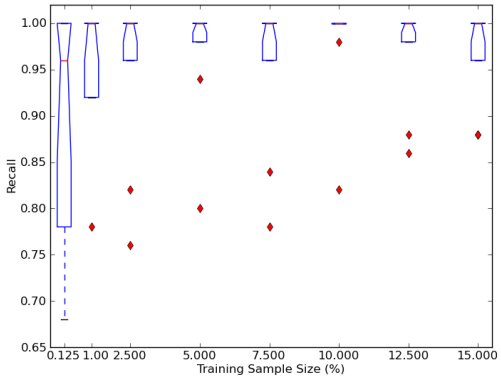


(b)

Figure 6: Box plots of precision (6a) and recall (6b) versus heatmap bin size. Top and bottom of boxes denote the 1^{st} and 3^{rd} quartiles, respectively; red band denotes median; whiskers represent 1.5 IQR; red diamonds denote outliers.



(a)



(b)

Figure 7: Box plots of precision (7a) and recall (7b) versus training set size measured as the percent of available sessions. Top and bottom of boxes denote the 1^{st} and 3^{rd} quartiles, respectively; red band denotes median; whiskers represent 1.5 IQR; red diamonds denote outliers.

lead required based on the distribution of the L_1 distances. Our preliminary analysis (not shown) suggests that beyond $k = 0.75$ we see only marginal improvement in precision and recall. Pending more in-depth evaluation, we set $k = 0.75$.

4.1.3 On Selecting the Training Set Size

Lastly, an important concern in building and deploying the traffic classifier is the amount of data required to build a training model for each protocol so that they are sufficiently stable and capture a broad enough variety of sessions to allow the classifier to reliably identify traffic belonging to that protocol. To explore this space, we held the confidence threshold and heatmap bin sizes constant ($k = 0$ and 5×5, respectively) and performed a set of 50 trials for each of various training sample sizes from 0.125-15% of the available data in the `dartmouth` dataset. The results are shown in Figure 7. Notice that the median precision and recall increase steadily as sample sizes increase. Beyond 10% there is only a marginal improvement; for the remainder of the paper we train the classifier on 15% of the available data.

4.2 Results

We now turn our attention to the performance of our classifier based on the aforementioned parameter settings. The confusion matrix generated by following the evaluation methodology outlined earlier is given in Figure 8. The matrix can be interpreted as follows: Each column represents the actual port from which testing samples are drawn; each row represents the port that the testing model was classified as belonging to. The color of the square at each row-column intersection intensifies with each decision that falls in that square. A completely black square, for example, indicates that such a classification never occurred in any of the trials, and a completely white square indicates that samples from the corresponding column were always classified in that manner. Thus, we would want to see a completely white line across the diagonal.

Notice that our accuracy is quite high, with the exception of confusion between traffic belonging to `MS Active Directory` service (445) and `NetBIOS` session service (139). The heatmaps for traffic on these ports in the `dartmouth` dataset is shown in Figure 5. Notice, however, how strikingly similar they are. Investigating this "misclassification," we find that, interestingly, traffic on these ports both relate to access to file shares, and are routinely attacked by similar malware. Moreover, in Windows NT/2000, the file sharing protocol is run over `NetBIOS`. Additionally, if the client has

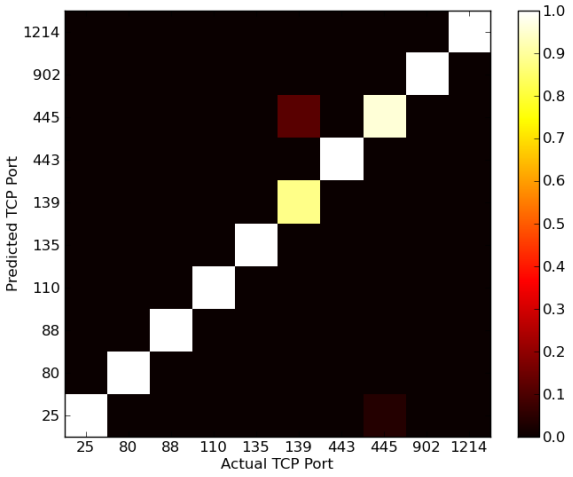


Figure 8: Confusion matrix for the `dartmouth` dataset.

`NetBIOS` enabled, it makes connections on both ports 139 and 445 simultaneously[3]. Without access to payloads, we cannot say with certainty what the cause of the confusion is. Nonetheless, we believe it underscores the success of our approach in that it provides a network administrator with a fast and reliable way to assess what types of traffic she may be seeing on a given port. Our average precision and recall were 98.5% and 98.4%, respectively.

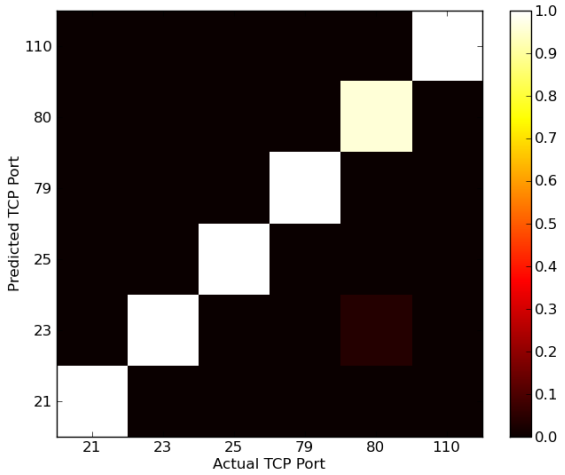


Figure 9: Confusion matrix the `darpa` dataset

Similarly, we provide results for a cross-validation test using the `darpa` dataset, with parameters chosen based on the previous experiments with the `dartmouth` data. Recall that in this case, we only have traffic from six protocols, so the evaluation only focuses on deciding if we can accurately classify one of these protocols. The results are shown in Figure 9. As before, the approach is very accurate, with our average precision and recall being 99.4% and 99.3%, respectively.

4.2.1 Inter-dataset analysis

[3]See `http://ntsecurity.nu/papers/port445/` for a more detailed discussion on this issue.

Lastly, to explore how well the models perform when training and testing on disparate networks, we perform the same type of cross-validation, but this time, train on samples selected from one dataset, and test on samples from the other. Specifically, we train on the 10 models from `dartmouth`, and take testing samples from the 3 protocols (`SMTP, HTTP, POP3`) for which we have ground truth in the `darpa` dataset *and* that are also in the `dartmouth` dataset. We then examine how well we do in classifying these protocols[4]. The results are shown in Figure 10. These experiments yielded an average precision and recall of 84.2% and 68.7%, respectively.

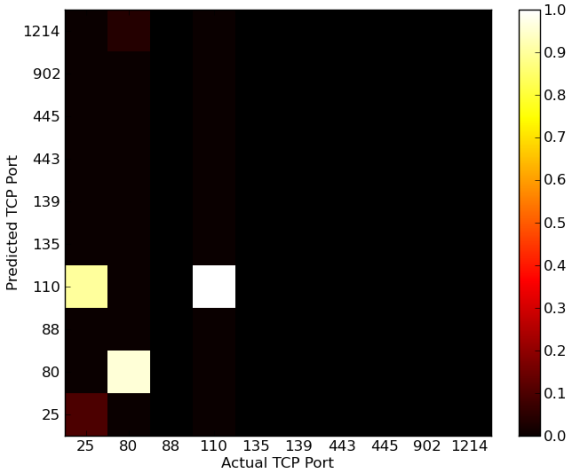
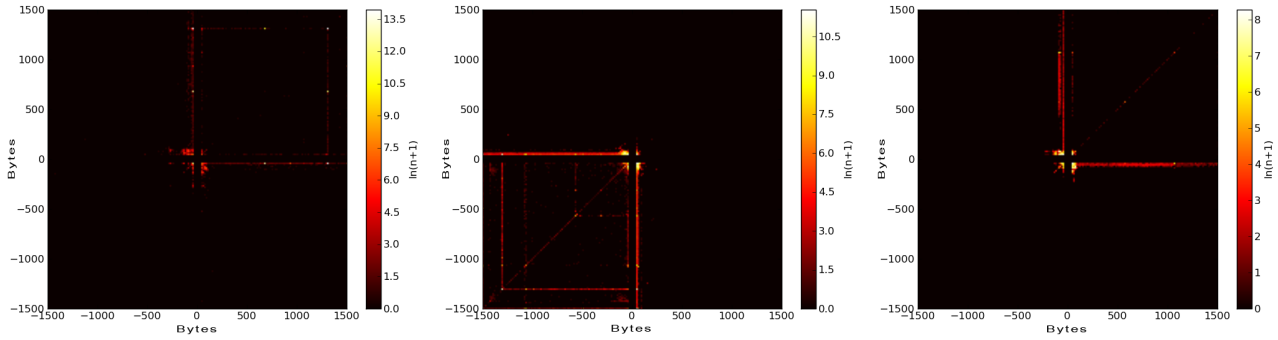


Figure 10: Confusion matrix for trials training on the `dartmouth` dataset and testing from the `darpa` dataset

The most notable misclassification is the confusion of port 25 (`SMTP`) and port 110 (`POP3`). Upon closer inspection of the data, we found a high prevalence of small packets exchanged in the `SMTP` data from the `darpa` dataset that is not present in the `dartmouth` dataset. We conjecture that this might be due to the presence of the so-called "Mailbomb" attack that is present in the `darpa` dataset. These high-intensity regions (see Figure 11) are more alike the pattern expected for `POP3`, which therefore mislead the classifier. Kendall [7] notes that the typical instance of this attack in the dataset is a series of 10,000 1-kilobyte email messages sent to a single user on port 25. We intuit that this attack would most certainly produce a high percentage of small-packet exchanges, as the short email message size would allow small TCP acknowledgements to become more dominant in the traffic pattern.

That said, this underscores the importance of training the classifier on data that is contextually relevant to the traffic that will be the subject of classification. In particular, it highlights a weakness of the approach, suggesting that in practice it would be prudent to build models from training data collect on the network to be monitored. A promising outcome, however, if that the results do show that the use of the visual motifs can be very helpful in classifying traffic and quickly spotting that which does not look like what one would expect.

[4]We found that due to discrepancies in the traffic patterns between the two datasets for the 3 protocols from which testing samples were drawn, a confidence threshold of $k = 0.75$ was too high; and would lead to the classifier not making any decisions. Therefore, we set $k = .25$ for this test.



(a) SMTP from dartmouth (b) POP3 from dartmouth (c) SMTP from darpa

Figure 11: Heatmaps for traffic on ports 25 and 110 in the dartmouth and darpa datasets

5. LIMITATIONS

A limitation of this work is its susceptibility to evasive maneuvers [1, 16], where a user attempts to disguise her actions by morphing or padding her packets to look like a different protocol. Techniques for mitigating such attacks remain as future work.

A more practical limitation is the fact that we assume that sessions can be separated from each other. Clearly, while this assumption holds for cleartext traffic or even for separating traffic employing application layer encryption (say using SSL), the assumption does not hold in cases where traffic from disjoint protocols is multiplexed over a single encrypted tunnel—as is typically done when using VPNs. Lastly, the approach we take is only applicable to TCP traffic. That said, since TCP dominates UDP traffic (e.g., 86% of the dartmouth data was TCP traffic), we argue this limitation may not be that significant in practice.

6. CONCLUSIONS & FUTURE WORK

In this paper, we demonstrate how a technique for heatmap visualization can be used to model application protocol behavior in TCP/IP traffic in addition to a method for using the L_1 distance to detect the presence of an application protocol using only those features that remain intact after encryption. We present a usage model wherein a network operator seeks to detect the presence of certain application protocols within a body of traffic. We evaluated our traffic classification methodology using a large dataset of real Internet traffic and found that our classifier performs well.

We are exploring how the classifier can be extended to work in an on-line fashion with the potential to adapt its training models using live data gathered during operation. We leave as future work an examination on how the techniques can be used to distinguish between different protocol behaviors of the same application (e.g, the file transfer vs. interactive mode of ssh, or streaming video and audio over http).

Acknowledgments

We thank the anonymous reviewers for their insightful comments and suggestions for improving the paper. This work is supported by the National Science Foundation under award CNS-0852649.

7. REFERENCES

[1] FOGLA, P., SHARIF, M., PERDISCI, R., KOLESNIKOV, O., AND LEE, W. Polymorphic blending attacks. In *Proceedings of the* 15^{th} *USENIX Security Symposium* (August 2006), pp. 241–256.

[2] GLANFIELD, J., PATERSON, D., SMITH, C., TAYLOR, T., BROOKS, S., GATES, C., AND MCHUGH, J. FloVis: Leveraging Visualization to Protect Sensitive Network Infrastructure. In *Symposium on Information Assurance and Cyber Defence (CATCH)* (march 2009).

[3] GOLDRING, T. 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* (New York, NY, USA, 2004), ACM Press, pp. 119–123.

[4] HORNIG, C. RFC 894: A Standard for the Transmission of IP Datagrams over Ethernet Networks, April 1984.

[5] JANIES, J. Existence Plots: A Low-Resolution Time Series for Port Behavior Analysis. In *VizSec '08: Proceedings of the 5th international workshop on Visualization for Computer Security* (2008), pp. 161–168.

[6] KARAGIANNIS, T., PAPAGIANNAKI, K., AND TSOS, M. F. BLINC: Multilevel traffic classification in the dark. In *ACM SIGCOMM* (August 2005).

[7] KENDALL, K. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's thesis, Massachusetts Institute of Technology, 1998.

[8] LAKKARAJU, K., YURCIK, W., AND LEE, A. J. 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* (2004), pp. 65–72.

[9] LIBERATORE, M., AND LEVINE, B. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the ACM conference on Computer and Communications Security* (October 2006), pp. 255–263.

[10] LIN, J., KEOGH, E., AND LONARD, S. Visualizing and discovering non-trivial patterns in large time series databases. *Information Visualization Journal 4*, 2 (April 2005), 61–âĂŞ82.

[11] MAIOLINI, G., BAIOCHHI, A., LACOVAZZI, A., AND RIZZI, A. Real Time Identification of SSH Encrypted Application Flows by Using Cluster Analysis Techniques. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference* (2009), pp. 182 – 194.

[12] MCHUGH, J. Testing Intrusion Detection Systems: A Critique of the 1998 and 1998 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and Systems Security 3*, 4 (2000).

[13] SONG, D., WAGNER, D., AND TIAN, X. Timing analysis of keystrokes and SSH timing attacks. In *Proceedings of the* 10^{th} *USENIX Security Symposium* (August 2001).

[14] SUN, Q., SIMON, D. R., WANG, Y.-M., RUSSELL, W., PADMANABHAN, V. N., AND QIU, L. Statistical identification of encrypted web browsing traffic. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 2002), pp. 19–30.

[15] TAYLOR, T., BROOKS, S., AND MCHUGH, J. NetBytes Viewer: An Entity-based Netflow Visualization Utility for Identifying Intrusive Behavior. In *Mathematics and Visualization (Proceedings of VizSEC)* (August 2008).

[16] WRIGHT, C., COULL, S., AND MONROSE, F. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the Network and Distributed Security Symposium - NDSS '09* (February 2009), IEEE.

[17] WRIGHT, C. V., BALLARD, L., COULL, S. E., MONROSE, F., AND MASSON, G. M. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy* (May 2008).

[18] WRIGHT, C. V., MONROSE, F., AND MASSON, G. M. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research 7* (December 2006), 2745–2769. Special Topic on Machine Learning for Computer Security.

[19] WRIGHT, C. V., MONROSE, F., AND MASSON, G. M. Using visual motifs to classify encrypted traffic. In *Proceedings of the* 3^{rd} *International Workshop on Visualization for Computer Security (VizSEC)* (November 2006).

[20] YIN, X., YURCIK, W., TREASTER, M., LI, Y., AND LAKKARAJU, K. Visflowconnect: netflow visualizations of link relationships for security situational awareness. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* (2004), pp. 26–34.

[21] YU-CHUN CHANG AND KUAN-TA CHEN AND CHEN-CHI WU AND CHIN-LAUNG LEI. Inferring speech activity from encrypted skype traffic. In *IEEE Globecomm* (2008).

Appendix A

Table 1 below provides insights on how we choose λ. It shows the average number of sessions that would be generated for the given port with the specified number of data points. Results averaged across 25 trials. Notice that at even at 25,000 data points, the median number of sessions is less than 1000.

	Testing Size				
Port	**10,000**	**15,000**	**25,000**	**35,000**	**45,000**
25	167.00	214.12	307.20	362.08	498.40
80	413.56	547.96	934.48	1190.80	1494.16
88	1553.40	2331.72	3885.44	5439.24	6993.72
110	413.28	575.80	973.00	1285.88	1689.24
135	1356.92	2033.68	3380.60	4729.04	6084.68
139	225.48	317.00	523.92	552.92	653.92
443	438.68	641.48	1027.00	1477.04	1844.96
445	30.12	28.16	65.56	95.84	116.28
902	1466.48	2211.56	3683.72	5131.04	6611.28
1214	31.92	37.32	65.12	69.76	86.44
median	413.42	561.88	953.74	1238.34	1591.70

Table 1: Mapping of average number of data points (per protocol) to number of sessions.