# In-the-Dark Network Traffic Classification Using Support Vector Machines

**William H. Turkett, Jr., Andrew V. Karode, and Errin W. Fulp**
Department of Computer Science
Wake Forest University
Winston-Salem, NC 27109

**Abstract**

This work addresses the problem of in-the-dark traffic classification for TCP sessions, an important problem in network management. An innovative use of support vector machines (SVMs) with a spectrum representation of packet flows is demonstrated to provide a highly accurate, fast, and robust method for classifying common application protocols. The use of a linear kernel allows for an analysis of SVM feature weights to gain insight into the underlying protocol mechanisms.

## Introduction

When an application is being used on a network, a flow of information is maintained between the client and server (or peer) computers involved in that application. Each flow is characterized by a specific data-sharing protocol, specified by the type of application in use. The back and forth messages in an application protocol are embodied at a lower level in a flow of packets between the computers involved. Each packet consists of headers - meta-information about the packet, such as content size, source address, destination address, destination port number, and error-codes - and a packet payload, the actual content being delivered. It is at this packet level that network traffic can be captured through network analysis tools.

The ability to classify network packet traffic by application is an important feature of modern network management tools, permeating problems such as network bandwidth allocation and security policy enforcement. Traditional approaches to traffic classification that examine payloads or port information suffer from several problems: the high computational cost of performing payload inspection, payload encryption, and the use of non-standard port numbers. This work addresses the problem of single flow in-the-dark TCP traffic classification, where *in-the-dark* indicates that the only parameters available are those that describe the session flow: features such as packet size, direction, and inter-arrival time. Support Vector Machines, in combination with a spectrum representation of flows, are used for classification. Experiments to evaluate performance make use of actual traffic traces and conditions used in other traffic classification studies.

## Previous Work

A number of approaches have recently been explored to address the in-the-dark network classification problem. Wright et al. (Wright, Monrose, and Masson 2006) use profile Hidden Markov Models to classify single-flow traffic. One model per protocol of interest is constructed, using two-match-state chains (for client and server packets) and insertion and deletion states to support resends, dropped-packets, and related traffic features. Bernaille and coworkers (Bernaille and Teixeira 2007) use the size and direction information for the first four to five data (non-control) packets of a connection. Training consists of clustering of labeled sessions, followed by labeling each cluster with the applications existing in that cluster. Classification of new flows uses heuristics to determine which cluster a flow is most likely to belong to and which label from that cluster is most appropriate. Early et. al. (Early, Brodley., and Rosenburg 2003) make use of decision trees for classification. Given a flow, packet windows of size *n* are collected and statistics on the use of TCP flags, mean size, and mean inter-arrival time are computed. Each of these observations is classified against a decision tree trained using the same size windows, and the classification and confidence value for that observation is recorded. The label for an entire flow is obtained by finding the class that has the maximal value after summing the confidence values obtained from each observation of the flow (each *n*-long window).

This work makes use of Support Vector Machines (SVMs) (Cortes and Vapnik 1995), an approach which supports non-linear binary classification. Provided with labeled training data, training an SVM involves finding the maximum-margin hyperplane that separates the two classes. If data is non-linearly separable in the input space, SVMs support translation of the data into a higher-dimensional space to find a separator, doing this efficiently by using the *kernel trick* to obtain necessary function values from the higher-dimensional space while working in the input feature space. The classifier that results from training is represented by a small number of training inputs called the support vectors, and classification of a new session requires summation over each support vector of the evaluation of a kernel function. A number of techniques to extend SVMs beyond binary classification have been considered, including combining outputs from multiple one-versus-one or one-versus-all classifiers (Allwein, Schapire, and Singer 2000).

If data is linearly separable in the input space, SVMs allow inspection of the weight function used for classification. The weight function can be used to determine which features are informative in classification, providing both insight into what features are important in separating protocols and allowing for re-training on a smaller feature set. SVM implementations support sparse feature representations, where only information about the features that a session has non-zero values for needs to be represented for that session.

This work characterizes a flow through the use of aggregate features as well as a spectrum representation of the flow. The spectrum kernel representation was suggested by Leslie et. al. (Leslie, Eskin, and Noble 2002) in research on protein sequence similarity. With respect to proteins, the spectrum representation uses one feature for each possible *k*-length subsequence of amino acids. A protein's representation using this feature set is a count of how many of each possible *k*-length subsequence appears when a sliding window of length *k* is passed over the protein. An interpretation of this model is that proteins that use *k*-length subsequences at similar rates are considered similar to each other. The idea maps directly to network flows, with packets in a flow being the equivalent of amino acids in a protein. Previous work has evaluated spectrum kernels in the security domain, with Yin et. al. (Yin, Tian, and Mu 2005) evaluating it relative to other string kernels in user masquerade detection.

## Methods

Given a network flow, the features listed in Table 1 are aggregated from the flow for a fixed number of packets.

Table 1: Aggregate Flow Features

| | |
|---|---|
| % URG packets | Whole flow |
| % ACK packets | Whole flow |
| % PSH packets | Whole flow |
| % RST packets | Whole flow |
| % SYN packets | Whole flow |
| % FIN packets | Whole flow |
| % unflagged packets | Whole flow |
| Total packet count | Whole flow,Client,Server |
| Average packet size | Whole flow,Client,Server |
| Average inter-arrival time | Whole flow,Client,Server |

Additional features are generated from the flow's spectrum representation. Each packet is labeled as one of six types, dependent on it's size and direction: 0 = client-small (cS), 1 = client-medium (cM), 2 = client-large (cL), 3 = server-small (sS), 4 = server-medium (sM), and 5 = server-large (sL). The client is defined as the session-initiating host. The three sizes map to packets with no data (ACK-only packets), with under 50 bytes of data, and with greater than 50 bytes of data. Given a *k*-length window and the six-state packet representation defined above, the spectrum representation adds $6^k$ additional features to the feature space of Table 1 (all base 6 numbers up to length *k*).

Figure 1 serves to justify our selection of packet sizes. It represents the packet-size distribution for the protocols of interest in this work from a published dataset. Packets of size 0 account for 46% of the data, packets with payloads under 50 bytes make up another 35%, and large packets constitute 19%.

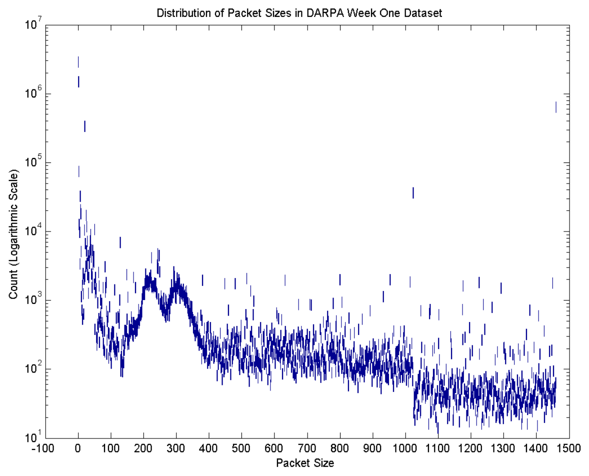

Figure 1: Packet size distribution for DARPA Week One dataset.

Table 2 depicts an example of a spectrum translation using a window of size 5 on 7 packets from an FTP connection. Computation of the spectrum feature is a straight-forward process that requires, during flow capture, keeping track of only the previous feature encoding and the state of the new packet. For the encoding below (windows of length 5 and 6 packet states), a feature encoding for the new window at time *t* can be computed from the encoding for the window at time *t-1* and the state of the new packet $s_t$ using the following formula: $f_t = (6 * f_{t-1})\%7776 + s_t$.

Table 2: Spectrum Representation

| Packet | Encoding | Spectrum |
|---|---|---|
| server, 89 bytes | 5 | Uses earlier packets |
| client, ACK only | 0 | - |
| server, 24 bytes | 4 | - |
| client, ACK only | 0 | - |
| client, 6 bytes | 1 | 50401 = Feature 6625 |
| server, 14 bytes | 4 | 04014 = Feature 874 |
| client, ACK only | 0 | 40140 = Feature 5244 |

Since network traffic classification requires differentiating between a number of different protocols, we use multiple one-versus-all binary SVMs. A classifier is trained for each protocol of interest, with known flows for that protocol labeled as positive examples and flows from all other protocols as negative examples. Classification requires evaluating a new flow under each trained SVM, and taking the maximal classification score as the protocol label to use.

A linear SVM kernel is used during training and classification. A linear kernel can be reduced from a support vector representation to a weight vector representation, allowing the sample score under a model to be computed as the dot product of the weight vector for a model and the feature vector of the new sample to classify. The successful use of a linear kernel brings with it the ability to directly investigate the weight vector for each model. Determining the maximum weight features can provide insight into the key components that delineate each underlying protocol. In addition, features that have zero weight across the models of interest can be removed.

## Results

Thorsten Joachim's SVMLight SVM implementation (Joachims 1999) was used in this work. Feature values were scaled between 0 and 1 in training, and testing data items were scaled using the bounds generated during training. A default regularization parameter, computed by SVMLight, was used in training each classifier.

### Traffic Classification

In this work, six protocols were evaluated: FTP, SSH, SMTP, HTTP, POP, and TELNET. Two datasets were used: attack-free data from the 1999 DARPA Intrusion Detection Evaluation Training Data Set (Lippmann and Haines 2000) and a Fall 2003 dataset from the Dartmouth CRAWDAD repository (Kotz, Henderson, and Abyzov 2004). This work assumes that a network dump tool is able to catch the initial packets of a session (packet 0 is the initial SYN packet). The number of examples of each type of protocol from each dataset is shown in Table 3.

Table 3: Available Protocol Data

| Protocol | DARPA Week 1 / 3 | Dartmouth |
|---|---|---|
| FTP | 439 / 461 | - |
| SSH | 73 / 71 | - |
| TEL | 713 / 649 | - |
| SMTP | 9780 / 10551 | 25 |
| HTTP | 205438 / 218277 | 469 |
| POP | 142 / 135 | 299 |

On data from Week One of the DARPA dataset, 10-fold cross validation was used to obtain an estimate of the ability of our approach to separate protocols and classify new samples correctly. Varying numbers of packets, collected sequentially from the start of a flow, were evaluated to determine how much data is required to obtain accurate classification rates. Figure 2 represents the accuracy of classification per protocol as the number of available packets increases. Length 5 windows were used for the spectrum encoding.

Because of the combination of high accuracy and low packet count requirement, this report uses 20-packet classifiers in further analyses. By viewing a confusion matrix for 20-packet classification, it is possible to evaluate the distribution of correct and incorrect classifications accumulated from the cross-validation process and to detect which protocols are mistaken for each other. When fewer packets are used, the causes of confusion shown in Table 4 are magnified, with the primary problems being the misclassification of TELNET as SMTP, and misclassifications in both directions between SMTP and FTP.

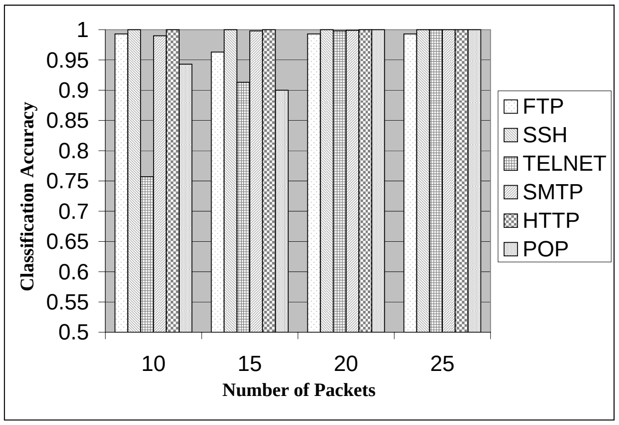

Figure 2: Classification accuracy with varying packet counts.

Table 4: DARPA Data Confusion Matrix (20 Packets, Aggregate and Spectrum Features)

| TYPE | FTP | SSH | TEL | SMTP | HTTP | POP |
|---|---|---|---|---|---|---|
| FTP | 436 | 0 | 0 | 3 | 0 | 0 |
| SSH | 0 | 73 | 0 | 0 | 0 | 0 |
| TEL | 0 | 0 | 712 | 1 | 0 | 0 |
| SMTP | 2 | 0 | 0 | 9778 | 0 | 0 |
| HTTP | 0 | 0 | 0 | 0 | 205438 | 0 |
| POP | 0 | 0 | 0 | 0 | 0 | 142 |

Ten-fold validation using 20-packet, 5-long window classifiers just using the spectrum representation (no aggregate features) showed a slight improvement in performance over the aggregate+spectrum classifier, as the two SMTP packets mis-classified earlier were classified correctly. In Table 5, the results of performing 10-fold cross validation on the same dataset using just aggregate flow features (those in Table 1) are reported. This execution was *naive*, mimicking the setup used in the previous cross-validation experiment: a linear SVM, default parameters suggested by SVMLight, and the same unbalanced dataset sizes. The results are poor, with the classifiers predicting most packets to be SMTP packets.

Table 5: DARPA Data Confusion Matrix (20 Packets, Aggregate Features Only)

| TYPE | FTP | SSH | TEL | SMTP | HTTP | POP |
|---|---|---|---|---|---|---|
| FTP | 0 | 0 | 0 | 439 | 0 | 0 |
| SSH | 0 | 0 | 0 | 73 | 0 | 0 |
| TEL | 0 | 0 | 0 | 713 | 0 | 0 |
| SMTP | 0 | 0 | 0 | 9780 | 0 | 0 |
| HTTP | 0 | 0 | 0 | 17 | 205421 | 0 |
| POP | 0 | 0 | 35 | 60 | 1 | 46 |

Table 6 presents results from performing 10-fold cross validation on 20-packet classifiers using different spectrum window sizes. Performance improves significantly once ordering information is included, as a window of size 2 performs much better than a size 1 window (which is equivalent to just computing the frequency of each of the six packet types). Results from window sizes 4 and larger remain consistent at over 99% correct classification for each protocol.

Table 6: DARPA Data % Correct (20 Packets, Aggregate+Spectrum, Varying Window Sizes [W])

| W | FTP | SSH | TEL | SMTP | HTTP | POP |
|---|---|---|---|---|---|---|
| 1 | 0.235 | 0.0 | 0.126 | 1 | 0.999 | 0.647 |
| 2 | 0.97 | 1 | 0.895 | 0.999 | 1 | 0.887 |
| 3 | 0.993 | 1 | 0.986 | 0.999 | 1 | 0.901 |
| 4 | 0.993 | 1 | 0.993 | 0.999 | 1 | 0.993 |
| 5 | 0.993 | 1 | 0.999 | 0.999 | 1 | 1 |
| 6 | 0.993 | 1 | 1 | 0.999 | 1 | 0.993 |
| 7 | 0.989 | 1 | 0.997 | 0.999 | 1 | 1 |

While the spectrum representation adds a significant number of attributes to be used in classification, and this increase brings with it a concomitant increase in memory and training time, we are interested in evaluating the effectiveness of the additional spectrum features for several reasons. There is a clear improvement in cross-validation performance when the additional spectrum features are included. Admittedly, tuning could be performed on the aggregate only classifiers, which is likely to lessen the performance difference. However, the earlier work of Early et. al. and Wright et. al. suggests there is still space for improvement in protocol classification, and we believe that describing ordered segments of flows may help bring about these improvements. We have demonstrated that cost of computing the spectrum representation for flows is relatively small, as each spectrum feature can be computed from the previous feature instead of having to be recomputed using the entire window length. Finally, as we will discuss under Feature Analysis, the used spectrum features end up being a fraction of the possible spectrum feature space.

To compare against the work of Early et. al. and evaluate the classifiers when a balanced number of training examples was presented, another experiment trained on 20-packet samples from the Week One DARPA data and tested on samples from Week Three of the DARPA data. The POP protocol was not tested in this set, as in Early's experiments. For training, fifty random samples per type were selected for a training set of 250 sessions and models for each of the five protocols were constructed. Five different testing sets, also composed of fifty random sessions per protocol, were classified using the trained models. The combined confusion matrix over these five testing sessions is Table 7.

To evaluate the generalizability of the classifier across datasets from different institutions, sessions from the Dartmouth dataset were classified using the 20-packet DARPA-trained classifiers. At 20 packets, SMTP and HTTP were classified correctly, but there was significant confusion with POP classifications primarily being classified as TELNET.

Table 7: DARPA Week Three Against Week One (20 Packets)

| TYPE | FTP | SSH | TEL | SMTP | HTTP |
|---|---|---|---|---|---|
| FTP | 244 | 0 | 3 | 3 | 0 |
| SSH | 0 | 250 | 0 | 0 | 0 |
| TEL | 0 | 0 | 250 | 0 | 0 |
| SMTP | 1 | 0 | 2 | 247 | 0 |
| HTTP | 0 | 0 | 0 | 0 | 250 |

We are currently evaluating the causes of this confusion and suggest mechanisms for improvement under Future Work.

Table 8: Dartmouth Against DARPA Confusion Matrix (20 Packets)

| TYPE | HTTP | FTP | SSH | TEL | POP | SMTP |
|---|---|---|---|---|---|---|
| HTTP | 469 | 0 | 0 | 0 | 0 | 0 |
| POP | 0 | 0 | 0 | 224 | 72 | 3 |
| SMTP | 0 | 0 | 0 | 0 | 0 | 25 |

Testing was also done to evaluate the response when a new protocol is presented to the system. Tests for this scenario were performed by iteratively training on five of the six protocols of interest and then introducing examples of the left-out protocol. Samples of the previously unseen protocol were commonly classified as matching a learned protocol instead of scoring poorly against all protocols, and this indicates another area of required future work. Table 9 presents the misclassifications for previously unseen examples from each type. Table 10 presents the portion of previously unseen examples where the maximal score was a positive score (minimizing this would be ideal, as all negatives scores are one potential indicator of a new protocol).

Table 9: Classifications for Unseen Types (20 Packets)

| TYPE | HTTP | FTP | SSH | TEL | POP | SMTP |
|---|---|---|---|---|---|---|
| HTTP | * | 0 | 0 | 0 | 36559 | 168879 |
| FTP | 0 | * | 0 | 189 | 0 | 250 |
| SSH | 0 | 0 | * | 0 | 0 | 73 |
| TEL | 5 | 266 | 0 | * | 202 | 240 |
| POP | 0 | 0 | 0 | 142 | * | 0 |
| SMTP | 44 | 5395 | 0 | 4156 | 185 | * |

Our evaluation of the time requirements for classification is currently focused on the SVM learning and classification phases. Model training times for the initial sets of experiments compare favorably against those reported in Early et. al. For the single-run 250 sample dataset, training times as reported by SVMLight were a cumulative 0.06 cpu-seconds. Training times averaged across the Week One cross-validation process, which required training on approximately 194,000 samples, were between 1.6 and 4.5 cpu-seconds per-protocol, with an average cumulative time of 16.82 cpu-seconds. Classification times averaged across the Week One cross-validation process, which required classifying 21,000 samples, were an average cumulative time of 0.07 cpu-seconds. Additional times that need to be taken

Table 10: Percentage Positive Scores for Unseen Types (20 Packets)

| HTTP | FTP | SSH | TEL | POP | SMTP |
|---|---|---|---|---|---|
| 0.005% | 98.4% | 100% | 50.2% | 100% | 92.4% |

into account in a final implemented system are the conversion of a stream of packets into an SVM flow representation, and the final classification step of finding the maximum over the returned per-protocol scores.

## Feature Analysis

Since a linear classifier was used, it is possible to extract the computed feature weights for each protocol. Evaluation of the feature weights can provide insights into the underlying features that delineate protocols, as well as can indicate features that are uninformative across all models.

To analyze feature weights, the 10-fold cross-validation mechanism used to test the effectiveness of the classifier algorithm was repeated five times, providing fifty different sets of weights per protocol. For each protocol, these output weight vectors were averaged to provide a single representation of the importance of features as measured over a large number of runs. In the models developed, most of the highly weighted features were spectrum features, indicating that the ordered flow of packets is informative. A total of 523 features were used across all protocols (a fraction of the possible feature space), with the following informative in each type of protocol: FTP: 187, HTTP: 477, POP: 315, SMTP: 423, SSH: 251, and TELNET: 427.

Using feature weights from the 20-packet classifiers using a spectrum representation with length-5 windows, a contextual translation of spectrum features in the top three weighted features for each protocol is listed in Table 11. In the table's description of a flow, cS, cM, cL, sS, sM, and sL retain their previously defined meaning. For some protocols, packet runs of size 6 or larger are shown when high weight features translated to overlapping windows.

By mapping the highly weighted features back to ASCII dumps of packet payloads, it can be determined that the highly weighted spectrum features appear to be picking up mainly two types of protocol behaviors: session setup (POP, HTTP, FTP, TELNET, SSH) and early session events (SMTP). For POP, the classifiers are detecting the opening of the session (cSsScS), followed by the server-side *OK POP3 server ready* and client-side *USER* communication. The HTTP classifier is detecting the initial client request and sharing of client information, such as Referrer and User-Agent. The FTP session features are detecting the initial server-side *220 Ready for new user*, client-side *USER*, and server-side *331 User OK* messages. The highest ranked TELNET feature is for packets sent primarily in the terminal setup phase. The SSH motifs represent an SSH protocol exchange between client and server, followed by the key exchange. The primary SMTP features appear tuned to detecting the MAIL, RCPT, and DATA exchanges, particularly the large DATA packet representing mail content.

Table 12 indicates relatively high-weighted non-spectrum

Table 11: High Weight Spectrum Features

| Encoding | Where Found |
|---|---|
| cSsScSsMcM | First five packets, 68% of POP |
| sMcMsSsMcSsS | Last six packets, 6% of POP |
| cScMsMcScL | Starts around packet 15, 2% of SMTP |
| sMcMsMcScL | Starts around packet 14, 2% of SMTP |
| cMsMcMsMcL | Starts around packet 12, 76% of SMTP |
| cSsScScLsS | First five packets, 78% of HTTP |
| cSsMcScMsL | Starts at packet 3, 31% of FTP |
| sLcScMsSsLcS | Starts at packet 4, 11% of FTP |
| cSsScScMsS | First five packets, 31% of TELNET |
| sMsScMsMsS | Starts at packet 8, 1 TELNET |
| sMcMsMsScM | Starts around packet 5, 1% of TELNET |
| cMsLcScLsSsM | Starts around packet 5, 100% of SSH |
| cScMsLcScL | cS before previous motif, 33% of SSH |

features. Only for the HTTP protocol were non-spectrum features listed in the top three highest weighted features. The highest weighted non-spectrum feature for POP is *% of FIN packets*, suggesting that an indicator of POP applications is fairly rapidly closing connections (appropriate for a mail checking protocol). The same argument holds for the high weight features of *% of SYN* and *% of FIN* packets for HTTP. HTTP and POP had the smallest average session lengths, 12 packets average for HTTP with 92% of sessions closed before 20 packets and 27 packets average length for POP and 31% of sessions closed before 20 packets. The highest weighted non-spectrum feature for SMTP is *% of PSH packets*. Informally, an IBM technical document (IBM 2008) suggests this is a common feature for SMTP servers, forcing the receiving end to start processing a long email as packets arrive instead of buffering the entire message. The importance of packet size for HTTP packets is relevant, as in the DARPA datasets HTTP packets are five times larger than the protocol with the next largest packet size (POP) and over thirty times larger than the packets of interactive protocols (TELNET, SSH).

Table 12: Highly Weighted Non-Spectrum Features

| Feature | Where Highly Weighted |
|---|---|
| % FIN packets | HTTP, POP, FTP |
| % PSH packets | SMTP |
| % SYN packets | HTTP |
| Average global packet size | HTTP |
| Client packet count | TELNET |
| % unflagged packets | SSH |

## Encrypted Traffic

Both the works of Bernaille and of Wright consider the problem of dealing with encrypted packet payloads. The primary effect this has on in-the-dark classifiers is a change in packet sizes. Packet sizes will be distributed differently because of block size changes upon encryption. To mimic encrypted traffic, Wright et al. rounded packet sizes up to multiples of 64 bytes to simulate the effects of a 512-bit block cypher. We evaluated our classifier after performing the same transformation on the DARPA Week One dataset. An updated *small packet* definition was used in generating the spectrum representation, from a maximum size of 50 bytes to 64 bytes. We performed 10-fold cross validation under this encryption model, using 20-packet classifiers. Results are reported in Table 13.

Table 13: DARPA Data Confusion Matrix (20 Packets, Simulated Encrypted Packets)

| TYPE | FTP | SSH | TEL | SMTP | HTTP | POP |
|---|---|---|---|---|---|---|
| FTP | 436 | 0 | 1 | 2 | 0 | 0 |
| SSH | 0 | 73 | 0 | 0 | 0 | 0 |
| TEL | 1 | 0 | 712 | 0 | 0 | 0 |
| SMTP | 1 | 0 | 0 | 9779 | 0 | 0 |
| HTTP | 0 | 0 | 0 | 0 | 205438 | 0 |
| POP | 0 | 0 | 0 | 0 | 0 | 142 |

The classifiers perform as well as those trained on non-transformed packet sizes. Because of the closeness of our original *small packet* designation to a 64 byte block, this change did not likely impact the models. The overall changes in average packet size have the potential to be significant, as average packet sizes for interactive sessions, where 1 byte packets are common, will increase by relatively large amounts. Since the packet size aggregate features were less significantly weighted than other features in our earlier classifiers, such effects were likely mitigated during classification.

## Discussion

The presented approach compares favorably with other approaches to in-the-dark traffic classification. All approaches discussed perform well on classification tasks. The advantages of the presented approach over related work: an increase in published performance (which we hypothesize is due to the spectrum representation); the use of only a portion of a session (matched in publication only by Bernaille); minimization of the number of models to compare against (fewer than Bernaille, equal to Wright, though more than Early), and an interpretable decision function via the linear separator (matched by Early's decision trees, where each branching rule translates to informative protocol information). Initial tests also indicate that the SVM approach is robust to changes in packet size due to encryption block ciphers, suggesting that encrypted packets can be supported as in the work of Wright and of Bernaille.

## Future Work

Additional effort into optimizing the training phases is needed, particularly an evaluation of the effects of tuning the regularization parameter in SVM learning. It is important to expand this work to include modern protocols of interest, such as instant messaging, P2P sharing, and VOIP. We are also interested in moving up a level of abstraction, to perform detection of content-type, such as text, media, voice, and video.

Determination of whether network classification models have to be trained locally or are abstract enough to be used at multiple sites is a very important problem. One possible cause of the reduced accuracy rates for Dartmouth network samples is an over-training on the DARPA dataset. Another possible cause is the role that network architectural features, such as transmission over wired vs. wireless networks, plays in packet characterizations. We intend to gather additional datasets both locally and from archives available on the web to analyze these issues.

Finally, we are particularly motivated to consider parallelization of the classifiers and linear kernel (weight vector) computations across processors for a live implementation.

## References

Allwein, E.; Schapire, R.; and Singer, Y. 2000. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, 9–16.

Bernaille, L., and Teixeira, R. 2007. Early recognition of encrypted applications. In *Proceedings of the Passive and Active Measurement Conference (PAM 2007).*

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20:273–297.

Early, J.; Brodley., C.; and Rosenburg, C. 2003. Behavioral authentication of server flows. In *Proceedings of the 19th Annual Computer Security Applications Conference*.

IBM. 2008. Ibm ztransaction processing facility. http://www-306.ibm.com/software /htp/tpf/maint/put19/pt1/PJ29832.htm.

Joachims, T. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*.

Kotz, D.; Henderson, T.; and Abyzov, I. 2004. CRAWDAD trace dartmouth/campus/tcpdump/fall03 (v. 2004-11-09). Downloaded from http://crawdad.cs.dartmouth.edu /dartmouth/campus/tcpdump/fall03.

Leslie, C.; Eskin, E.; and Noble, W. 2002. The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, 566–576.

Lippmann, R., and Haines, J. 2000. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, 162–182.

Wright, C.; Monrose, F.; and Masson, G. 2006. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research* 7.

Yin, C.; Tian, S.; and Mu, S. 2005. Using gap-insensitive string kernel to detect masquerading. In *Advanced Data Mining and Applications*, 323–330.