HMM Profiles for Network Traffic Classification (Extended Abstract)

Charles Wright
Johns Hopkins University
Information Security Institute
cwright@cs.jhu.edu

Fabian Monrose
Johns Hopkins University
Information Security Institute
fabian@cs.jhu.edu

Gerald M. Masson
Johns Hopkins University
Information Security Institute
masson@jhu.edu

ABSTRACT

We present techniques for building HMM profiles for network applications using only the packet-level information that remains intact and observable after encryption, namely, packet size and arrival time. Using less information than previously thought possible, we demonstrate classification accuracy close to that of other recent techniques, and show success in classifying a variety of common network applications as observed from real Internet traffic traces.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Miscellaneous; G.3 [**Probability and Statistics**]: Time series analysis, Markov processes; I.6.5 [**Model Development**]: Modeling methodologies

General Terms

Security, Measurement

Keywords

masquerade detection, intrusion detection, behavioral modeling

1. INTRODUCTION

Over the last several years, timing and size information have proven to be surprisingly powerful in identifying patterns in encrypted traffic. Such information has been used, for example, for inferring the web pages being accessed over `SSL` [18], for detecting stepping-stones [20], and for recovering keystroke information over `SSH` traffic [17]. Our primary goals are to investigate the feasibility and limits of similar techniques on more general forms of encrypted network communication including `SSH` tunnels and virtual private networks.

VizSEC/DMSEC'04, October 29, 2004, Washington, DC, USA.

Performing such traffic analysis requires, at the least, accurate models for common network protocols, using no more information than a packets' size, timing and direction. Here, we present our first attempt at building such a model and demonstrate our early success in applying our models to application level profiling for the purposes of protocol identification — a primitive form of masquerade detection. Specifically, we apply profile Hidden Markov Model techniques from [10, 6] to perform multiple sequence alignment. Flows that align "close" to one another, i.e., those that have long subsequences in common or that have very similar subsequences, are viewed as being related in the sense that they probably indicate instances of the same protocol.

The use of HMM methods is especially appealing for protocol detection for a few reasons. First, unlike most other sequence alignment techniques, they can generate an alignment for a large number of sequences without first calculating all pairwise alignments. For our application, this is particularly important as it means that we can train the HMM on hundreds of traces of the same protocol and generate a profile for the protocol without human assistance. HMM methods have also been shown to outperform more traditional methods for multiple sequence alignment in scenarios involving the frequent insertions and deletions that occur when aligning similar sequences with many small differences [6] (which is likely the case when comparing sessions in network packet traces). Another attractive feature of the profile HMM method is the ease with which modifications can be made to recognize shorter patterns within longer sequences.

We note that the technique we have pursued thus far relies on application protocols exhibiting consistent and observable structure and patterns in the series of packets they emit. Most application layer protocols do have such structure, which is largely defined by RFC specifications. Exceptions to this rule include human-driven traffic like `Telnet` or Secure Shell (`SSH`), where most messages are the results of user action and not solely a protocol definition in an RFC. Moreover, an active adversary could defeat our traffic analysis by closely controlling his sending rate and carefully crafting the size of the packets he sends. In fact, all of the challenges related to successful stepping stone detection [20] also apply here.

The rest of this paper is organized as follows. In Section 2 we review related work on masquerade detection, sequence alignment, and network traffic modeling. Section 3 discusses work on packet measurements within single flows on a wide-

area network link and relevant statistics for feature selection when building suitable models. Section 4 explains the design of our Hidden Markov Model. We present our experimental setup in Section 5 and empirical results in Section 6. Lastly, we present conclusions and discuss future directions in Section 7.

2. RELATED WORK

Much of the recent research on the application of principles and techniques from Biology to Computer Security [11, 2] borrows ideas from the work of Forrest *et al.* [8, 7]. For example, [8] introduces a technique based on concepts from immunology to detect intrusions in Unix™processes that examines n-grams of system calls issued by the processes in order to infer anomalous behavior. Lane and Brodley [12] subsequently improve on this technique and explore the use of more sophisticated sequence matching algorithms for comparing new n-grams to those previously observed.

Similarly, Ye [19] uses Markov chains to model Solaris audit events. A fully-connected model is derived where each possible event is represented by its own state, and each state delimited with nonzero transition probabilities to every other state in the model. As is the case with [8, 12], Ye's system operates on n-grams of events, but instead uses the Markov model to calculate each n-gram's probability. When the probability of an n-gram is too low, the approach considers such an event as an intrusion. A similar idea was also suggested by DuMouchel [4] for inferring a users' shell commands [4].

More closely related work is that of Early *et al.* [5]. Using n-grams of packets, a decision tree classifier was developed capable of distinguishing among flows from `HTTP` , `SMTP` , `FTP` , and `Telnet` servers with accuracy ranging between 85 and 100 percent.[1] Features included in the decision tree include TCP flags, average inter-arrival time, and average packet size. Our approach is similar to that of [5], but differs in a few important ways. First, we build models based on each packet's characteristics instead of using average values over n-grams of several packets. Second, their classifier uses data from TCP headers, while ours uses only the information that remains intact and observable after encryption, that is, packet size and arrival time. Additionally, while their decision tree algorithm uses information gain estimation to automatically select the best features, we use only one predetermined feature in each of our models. Surprisingly, this latter simplification still yields comparable accuracy.

3. MEASUREMENTS

Given that our initial target is to build models for packet flows in wide-area network traffic, a solid understanding of how these flows look and behave is paramount. In particular, it remains important that we assure that our features of interest are sufficient to distinguish the protocols from each other. To gain a better understanding of such traffic's characteristics, we briefly present a short series of measurements using traffic captured from George Mason University's Internet link in the fall of 2003. We examine these two features by following a similar analysis to that performed by Karagiannis *et al.* [9] on aggregate traffic in the network core.

[1]It is unclear from [5] whether these empirical results are based on LAN or WAN traffic.

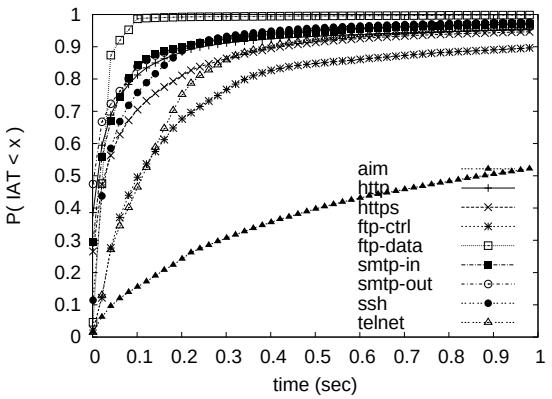

Figure 1: Cumulative Distribution of Packet Inter-arrival Times

There it was found that inter-arrival times are independent and follow an exponential distribution at sub-second time scales, and that consecutive packets' sizes are independent of each other and independent of the associated inter-arrival times. We find that most application-layer protocols differ from aggregate traffic in these regards.

Features. Figure 1 depicts the CDF of the inter-arrival times. It is clear that, for non-interactive traffic, inter-arrival times do seem to follow an exponential distribution. However, the interactive protocols do not seem to be as close to an exponential distribution as the machine-generated protocols; the tails of their distributions are too heavy to be exponential, so perhaps these distributions are better explained by a log-normal distribution or Pareto distribution as suggested in [14]. We plan to analyze the true distributions of this data set in the full version of this paper. Moreover, we calculate the autocorrelation function (ACF) of the packet inter-arrival times at several lags (see Figure 3). Essentially, if the times are independent, the ACF will be very close to zero, but an ACF of zero does not necessarily imply independence. Since we are dealing with a collection of thousands of sequences for each protocol, and not a single stream of packets as with aggregate traffic, we look at the average ACF over all instances of each protocol. (Note that this should not over-estimate the true ACF for the protocol, since some packet sequences for a protocol may be positively autocorrelated, while others may have a negative correlation and so cancel out towards zero.)

Observations

In [9] it was noted that the sample ACF for inter-arrival times in a trace of aggregate traffic was below 0.05 for almost all lags. Figure 3 shows that, in the traffic we analyzed, all application-layer protocols exhibit significant autocorrelation in their inter-arrival times for several lags. Similarly, the log of the inter-arrival times in Figure 3 all argue against independence. Therefore we conclude that the inter-arrival times and their corresponding log values may both be acceptable features for modeling flows. We build models to explore this possibility for log inter-arrival times and present our results in Section 6.

Karagiannis *et al.*[9] also observed the ACF of packet sizes over several lags and noted that the sample ACF for packet

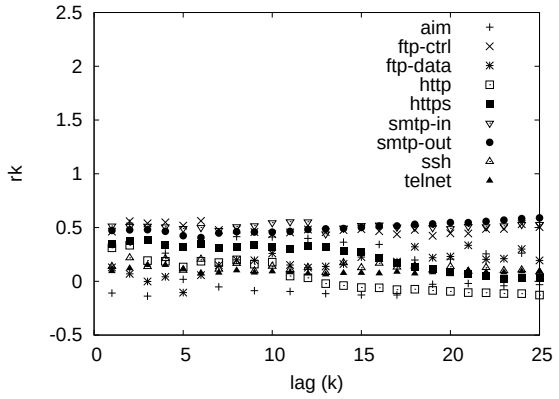

Figure 2: Average ACF for Packet Inter-arrival Times

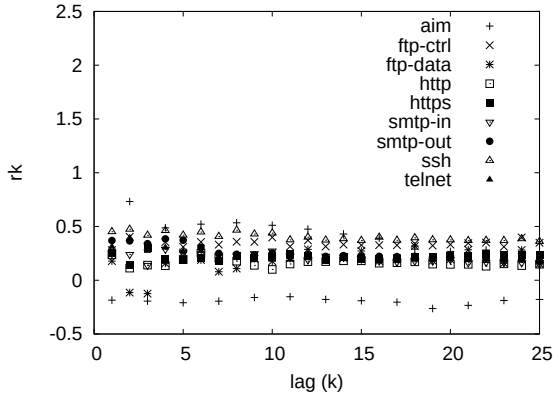

Figure 3: Average ACF for the Logs of the Inter-arrival Times

sizes was almost always below 0.05. The average ACF for each lag is given in Figure 3. For the most part, these values also are significantly higher than the 0.05 reported for aggregate traffic leading to the conclusion that the sizes of packets within the same session are clearly related. Based on the high autocorrelation coefficients at lag 1 for several protocols, we expect packet sizes to be a good feature for building HMMs. We explore the power of packet size as a protocol discriminator in the subsequent sections.

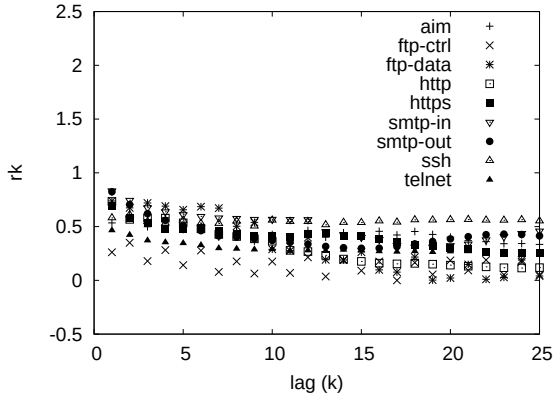

Figure 4: Average ACF for Packet Sizes

4. HMM STRUCTURE

In this section we describe the structure of our profile Hidden Markov Models. Specifically, we build and analyze two types of models for network flows. Each considers only a single feature for packets in the flow: the first model uses a packet's size to represent the packet, and the second uses inter-arrival time. Both are built on the same basic structure.

Our design closely follows the general profile model used for protein sequence alignment, which can best be described as a left-right model built around a chain of hidden states (namely, one for each position in the chain of amino acids that make up the protein). Each state emits symbols with a probability distribution specific to its position in the chain. States in this central chain are referred to as `Match` states, because their probability distributions for symbol emissions match the normal structure of the protein family. To allow for variations between proteins in the same family, the model has two additional states for each position in the chain. One, called `Insert`, allows for one or more extra amino acids "inserted" in an otherwise conforming sequence, between two normal parts of the chain. The other, called the `Delete` state, allows for a position to be omitted from the sequence. In the context of network flows, the `Insert` states represent duplicate packets and retransmissions, while the `Delete` states allow the model to account for packets lost in the network or dropped by the detector. Both types of states may also represent other protocol-specific variations in higher layers of the protocol stack.

The main difference between the structure of our HMM (see Fig. 5) and that used for [protein] sequence alignment is that we have separated the chain of `Match` states into two parallel and interconnected chains of matching states. The states in one chain match packets from the client, while those in the other match packets from the server. This change is necessary as the distribution of packets seen in a session is not solely dependent on the packet's position within the session. In Section 3, we show that packets are likely to also be dependent on previous packets in the session. As the Markov property requires that the current state depends only on the previous state, to correctly express this interdependence on previous packets, we therefore need more than one `Match` state per position.

Our two-match-state design is a compromise between having one match state per column (as used in most biological models) and having a match state for every symbol in each column—essentially having one of the earlier fully-connected models as each of our columns. The former is extremely efficient and simple, but may not characterize network traffic well. The latter sounds like a wonderfully faithful model of the traffic, but the number of states and state transitions required would make parameter estimation painfully slow.

Again, we note that for our technique to be successful, it requires that application protocols exhibit consistent and observable structure and patterns in the series of packets they send. While this may be the case for some protocols, example `scp`, this is not always so. A prime example of a protocol that generates significant free-form traffic is `Telnet`, where most messages are a result of user action, and not from interactions defined by the RFC. As we show

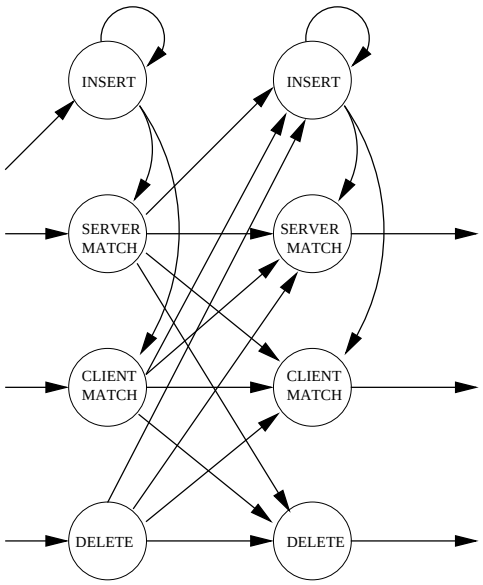

Figure 5: The Basic Structure of Our Two-Match-State Profile HMM

later our current classifier does not perform as well under such conditions.

5. EXPERIMENTAL PROCEDURE

We experimental evaluate our approach using traffic from two distinct sources, namely (*i*) packet-level traces collected at George Mason University in August and September of 2003, and (*ii*) packet traces from the MIT Lincoln Labs Intrusion Detection Evaluation [13]. The latter traces were used primarily to facilitate a more direct comparison with earlier work.

From the GMU traces we selected a week at random and examined traffic over a 10-hour period between 8 a.m. and 10 p.m. each day. Details of the capture process beyond our control dictate that the captures only include traffic from the first 10 minutes of every quarter hour. We consider every active flow at the beginning of a 10-minute increment to be the start of a new flow, since we do not have sufficient information to determine the events in the 5 minutes between captures. We extract TCP sessions from Internet clients to hosts in the GMU network on the well-known ports for `HTTP` (80), `HTTP` over `SSL` (443), `FTP` control (21), `FTP` data (20), `SMTP` (25), `Telnet` (23) and `SSH` (22). Additionally, we extract outgoing `SMTP` sessions, since these are also quite likely to involve internal servers. Additionally, we include AOL Instant Messenger (`AIM`) traffic as another example of an interactive protocol with human-generated events. All sessions consisting of less than 5 packets are omitted from our analysis.

To build and train our models, we make use of the General Hidden Markov Model Library [16], developed by the Algorithmics group at the Max Planck Institute for Molecular Genetics. To build an application profile HMM, we first manually construct an initial model in which all packets are equally likely in every position in the sequence. We randomly select 400 flows of the given type and use them to re-estimate the initial model's parameters, using the Baum-Welch [1] version of the EM [3] algorithm. Then, to classify a sequence, we compute its probability of being generated by each of the models, and assign to it the label of the model that generates it with the highest probability. All classification results presented here are the average classification percentages, calculated over at least 5 runs. We generate a confusion matrix for each classifier's results, to illustrate not only how well our classifier and models work, but also show how and when they fail. To the standard confusion matrix we report an extra parameter that depicts the percentage of flows for each protocol that failed to match any model.

Protocol	Classifications (percent)				
	ftp	smtp	telnet	http	none
ftp	99.4	0.0	0.0	0.0	0.6
smtp	0.1	97.9	1.0	0.0	1.0
telnet	3.1	0.0	21.7	1.6	73.6
http	0.0	0.1	0.5	96.9	2.4

Table 1: Confusion Matrix for the Size-Based Classifier on MITLL traces (blocksize = 64 bytes)

As discussed earlier, packet sizes appear to be a good feature to use for building traffic profiles—however, if encryption is being used, we will not be able to see the true size of the cleartext packet. Instead, only payload sizes rounded to the next whole number multiple of the cipher's block size will be observed. Therefore, we compensate for this by rounding the observed packet sizes up to the next whole multiple of the block size, and build profiles for packet sizes using a handful of different block sizes.

Similarly, while packet inter-arrival times may be a good feature for building traffic profiles, we argue that the logs of the inter-arrival times may be more suitable. The heavy tails on some of the protocols' CDFs in Figure 1 illustrate the wide range of values observed for inter-arrival times. To build a HMM using that data would require the use of continuous variables or a discrete model with an unwieldy number of symbols. However, since the packet capture process used to record these traces requires that all inter-arrival times fall between 0 and 10 minutes, we know that the log of the inter-arrival time in seconds is also bounded. Consequently, as HMMs for continuous variables are much slower and more complex to work with, we quantize this range and build a discrete-density HMM [15]. Of course, some information is lost due to quantization, so to better evaluate the effect of quantization on classification accuracy, we report results for different sampling rates.

6. EMPIRICAL RESULTS

We first present the results of our two classifiers, examining `FTP`, `SMTP`, `HTTP`, and `Telnet` sessions using the data from the MIT Lincoln Labs Intrusion Detection Evaluation [13]. We do so in order to provide a quick comparison with the results in [5].

SIZE-BASED CLASSIFIER**.** Our results in Table 1 show that, except for `Telnet`, the accuracy of our packet size-based classifier is at worst 3 percent below the best Early *et al.* results on traces from the same dataset. For `SMTP`, our accuracy is over 10 percent higher. Since the decision tree method in [5] uses information gain estimation to automatically select the best features for classification, we find it somewhat sur-

Protocol	Classification Probability (percent)									
	aim	smtp-out	smtp-in	http	https	ftp-data	ftp	ssh	telnet	none
aim	78.1	2.7	0.9	0.6	1.7	0.3	0.3	2.5	1.9	11.0
smtp-out	2.5	70.2	8.0	0.2	0.7	0.3	2.1	0.9	5.0	10.2
smtp-in	1.1	11.4	73.3	0.1	0.7	0.3	0.5	1.7	6.0	5.1
http	0.2	0.2	0.2	75.8	6.5	5.9	0.0	1.8	0.8	8.4
https	0.4	3.0	1.9	2.2	68.1	1.5	0.0	3.5	2.1	17.2
ftp-data	1.8	6.6	4.3	13.9	6.2	54.7	0.2	2.3	4.1	6.0
ftp	1.4	19.6	4.7	0.2	0.0	1.7	63.8	4.1	4.3	0.1
ssh	2.6	1.5	1.2	0.9	4.7	0.5	1.8	40.8	0.7	45.4
telnet	3.0	3.6	3.4	2.6	6.0	6.9	2.4	4.7	30.2	37.2

Table 3: Confusion Matrix for the Size-Based Classifier on GMU Traces (blocksize = 16 bytes)

Protocols	Classifications (percent)				
	ftp	smtp	telnet	http	none
ftp	82.2	0.0	0.0	0.0	17.8
smtp	2.5	96.6	0.0	0.2	0.7
telnet	1.3	2.6	14.2	0.0	81.8
http	0.3	0.8	0.3	95.3	3.3

Table 2: Confusion Matrix for the Timing-Based Classifier on MITLL traces (sampling rate = 1)

prising that our single-feature classifier, using inexact packet sizes, performs so well on most protocols. It appears that, for noninteractive network flows, packet sizes are indeed a very good indicator of the protocol in use.

Similarly, Table 2 shows the accuracy of our timing-based classifier on traces from MITLL. Again, our model does not characterize the `Telnet` flows well at all, but accuracy on the other protocols is within 5 percent of the results in [5]. Our classifier and theirs have difficulty with different protocols (`FTP` and `SMTP`, respectively) but the penalty seems to be about the same in both cases. In some cases, our misclassification rates are actually lower than those in [5] because our classifier has the option of reporting a "don't know" condition for flows that are extremely unlikely for all of the models, while the decision tree classifier must always return some classification. For noninteractive flows, packet interarrival times also appear to be quite good indicators of the protocol in use.

For the rest of this section, we focus on classifying the larger mix of protocols in the more realistic (and challenging) GMU data. Tables 3 and 4 show the results of our size-based classifier using typical block sizes of 16 and 32 bytes, respectively. Interestingly, we find that our classifier improves in accuracy in many cases even with block sizes much larger than that likely to be seen in practice (e.g., 256-byte blocks).

If we allow for confusions of the different directions of `SMTP`, we see that classifying flows from nine protocols using only a single feature works almost as well as classifying flows from only four using a decision tree. We generally classify `AIM` and `HTTP` most accurately. The most common error is to confuse `FTP` sessions with outgoing `SMTP`. Early *et al.* noted a similar effect with their classifier, where very long `SMTP` flows tend to be mistaken for either `FTP` or `Telnet`. It was suggested that the confused `SMTP` sessions, on the wire, do "look" very much like `FTP` or `Telnet`. For our classifier, decreasing the block size (thereby increasing the precision of the data) tends to decrease the error rate. For example, Table 4 shows `FTP` misclassified as outgoing `SMTP` 22.7 percent of the time when using a 32-byte block size; this rate decreases to 19.6 percent when we lower the block size to 16 bytes in Table 3. As one might expect, given interactive traffic's relatively unstructured nature and the fact that our `SSH` dataset contains both `SSH` and `SCP`traces, this classifier generally performs the worst on `Telnet` and `SSH`.

The most surprising result is for AOL Instant Messenger (`AIM`) traffic. Because the Instant Messenger is an interactive, human-driven application, like SSH and Telnet, we might expect that our classifier would not classify it accurately. However, to the contrary, we find that both of our classifiers consistently give their best results when examining `AIM` sessions. A visual inspection indicates that most of the packets in the `AIM` traces are the result of machine-driven communication between logged-in clients sitting idle and AOL's servers, and not the result of human users involved in conversations.

Time-based Classifier. Tables 5 through 7 show the results of our time-based classifier when used with different sample rates. We note that information loss (in this case due to quantization) does not always hurt our overall accuracy. Compared to our results with the size-based classifier, the biggest change appears to be for `FTP`, where our accuracy drops by about 10 percent overall. Interestingly, instead of being misclassified as outgoing `SMTP`, `FTP`now gets confused more frequently with `SSH`. This effect tends to decrease as our sampling rate increases, so we see that, while an increased sampling rate does not seem to help our overall accuracy, it does tend to reduce the most common errors. At these sampling rates, the trend is not as strong as it is for the size-based classifier, but we can nevertheless observe a drop in common errors as the sampling rate exceeds 5. For example, the rate of confusions of `FTP` as `SSH` falls from 14.3 percent in Table 6 to 12 percent in Table 7.

Since many `FTP` control connections are human-driven (although likely through a web browser or other graphical interface), it's logical that the sizes of the commands issued are more easily confused with `SMTP`, which has a similar "numeric code and status message" format, in a model based on packet sizes, and easily confused with an interactive protocol by a model built on the time between commands. With both classifiers, increasing the precision of our data decreased the rate of confusions.

	Classification Probability (percent)									
Protocol	aim	smtp-out	smtp-in	http	https	ftp-data	ftp	ssh	telnet	none
aim	81.6	1.9	0.9	0.4	2.5	0.5	0.5	3.4	2.7	5.6
smtp-out	4.0	65.6	12.7	0.0	0.6	0.3	7.2	0.8	4.4	4.4
smtp-in	1.1	11.4	70.2	0.0	0.3	0.2	2.0	0.6	11.9	2.3
http	0.2	0.1	0.1	81.1	6.2	7.5	0.1	0.6	0.8	3.6
https	0.5	4.1	0.2	2.8	76.5	1.9	0.1	2.2	3.2	8.5
ftp-data	2.1	4.4	5.1	12.1	4.0	62.7	0.2	0.4	5.9	3.1
ftp	1.0	22.7	2.9	0.1	0.0	2.0	62.7	3.9	4.7	0.0
ssh	7.0	1.4	0.7	1.1	14.7	0.4	2.5	42.0	1.9	28.1
telnet	2.6	4.1	1.0	4.4	8.0	4.7	2.4	5.4	42.0	25.5

Table 4: Confusion Matrix for the Size-Based Classifier on GMU Traces (blocksize = 32 bytes)

	Classification Probability (percent)									
Protocol	aim	smtp-out	smtp-in	http	https	ftp-data	ftp	ssh	telnet	none
aim	86.4	0.4	0.6	0.5	1.7	0.1	1.7	3.9	1.8	2.7
smtp-out	2.3	66.8	9.4	1.6	2.3	0.4	2.9	4.3	4.8	5.0
smtp-in	2.3	10.2	67.1	2.0	3.7	1.5	3.3	6.2	1.6	2.0
http	1.3	3.0	1.9	67.6	10.9	7.0	2.2	2.8	0.7	2.4
https	1.1	3.3	2.9	12.4	56.0	4.9	2.8	7.6	2.1	6.9
ftp-data	0.0	2.4	5.5	13.7	6.9	61.6	1.1	2.8	1.4	4.5
ftp	4.0	3.9	5.9	0.8	2.9	5.5	49.8	13.3	9.2	4.9
ssh	10.6	1.8	3.8	1.6	5.2	1.9	4.1	29.0	9.7	32.4
telnet	6.2	2.8	2.1	0.6	3.2	0.8	6.5	13.0	31.7	33.1

Table 5: Confusion Matrix for the Timing-Based Classifier on GMU Traces (sampling rate = 3)

7. CONCLUSIONS AND FUTURE WORK

We present two new single-feature classifiers for network flows, based on profile Hidden Markov Models. We show that even imprecise measurements of single per-packet features often characterize network flows surprisingly well. Moreover, our empirical results demonstrated accuracy close to that of other recent approaches, but unlike earlier approaches, show some promise in classifying wide-area traffic.

In future work, we intend to explore fuzzy k-means clustering [10, 16] for packet sequences, using HMMs as cluster centers. With this technique, we hope to be able to better distinguish between `SSH` and `SCP` traffic. Our early observations show that there are two very distinct populations of sessions within our dataset for port 22 (`SSH`). When one type dominates the set of sequences used to build the model, almost all sequences of the other type fail to classify. Likewise, when the other sequences dominate the model, the other traces seem to classify as almost anything but `SSH`. To better separate these sequences, we plan on using an HMM clustering algorithm, with profiles for `FTP` data and `Telnet` as the initial cluster centers. The hope is that `SSH` sessions would cluster around the interactive `Telnet` profile, and `SCP` sessions would cluster around the `FTP` bulk data transfer profile. At the end of the procedure, the new cluster centers should give us profiles for `SSH` and `SCP`. Lastly, we expect to evaluate the performance of a HMM for two variables, built on both packet sizes and logscale inter-arrival times, which we suspect might outperform our current models.

8. ACKNOWLEDGMENTS

The authors would like to thank Avi Rubin and David Marchette for their encouragement and guidance. We are also grateful to the Statistics group at GMU for sharing their packet traces and computational resources. Special thanks to the anonymous reviewers, whose suggestions significantly helped the presentation of this paper.

9. REFERENCES

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, February 1970.

[2] S. Coull, J. Branch, B. Szymanski, and E. Breimer. Intrusion detection: A bioinformatics approach. In *Proceedings of the* 19^{th} *Annual Computer Security Applications Conference*, pages 24–33, December 2003.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[4] W. DuMouchel. Computer intrusion detection based on bayes factors for comparing command transition probabilities. Technical Report 91, National Institute of Statistical Sciences, February 1999.

[5] 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. IEEE, December 2003.

	Classification Probability (percent)									
Protocol	aim	smtp-out	smtp-in	http	https	ftp-data	ftp	ssh	telnet	none
aim	84.2	1.0	1.0	0.6	1.4	0.2	0.8	2.8	1.2	6.9
smtp-out	2.9	66.1	10.3	1.4	2.2	0.6	2.1	4.7	2.9	6.7
smtp-in	2.1	11.6	64.0	2.0	4.2	1.3	3.2	6.3	1.9	3.4
http	0.9	4.8	3.6	62.2	11.3	6.2	2.0	3.9	0.7	4.6
https	1.2	3.7	3.7	9.8	51.9	3.8	3.7	8.7	1.9	11.8
ftp-data	0.3	3.3	7.2	11.9	5.2	58.9	0.8	5.4	1.4	5.4
ftp	5.1	4.6	7.7	1.3	3.9	3.4	47.3	14.3	5.7	6.8
ssh	9.8	2.3	3.7	1.5	4.4	1.8	2.2	26.7	8.1	39.2
telnet	5.8	3.1	2.5	0.5	2.2	0.6	4.8	13.9	23.9	42.8

Table 6: Confusion Matrix for the Timing-Based Classifier on GMU Traces (sampling rate = 5)

	Classification Probability (percent)									
Protocol	aim	smtp-out	smtp-in	http	https	ftp-data	ftp	ssh	telnet	none
aim	80.2	0.7	0.7	1.6	1.0	0.3	0.8	3.0	1.0	10.6
smtp-out	2.7	64.3	9.2	1.5	2.2	0.4	2.2	3.6	2.9	11.1
smtp-in	1.9	9.5	66.0	2.5	4.6	1.1	1.8	5.0	1.2	6.5
http	1.0	4.8	5.7	56.5	14.3	5.1	2.0	2.8	0.6	7.1
https	1.4	3.6	4.8	10.1	52.3	3.1	3.0	5.5	1.5	14.7
ftp-data	0.1	3.3	6.3	13.1	7.6	55.2	1.1	4.9	1.2	7.3
ftp	4.9	5.1	8.5	1.8	5.3	3.1	43.5	12.0	6.4	9.4
ssh	8.7	2.7	3.6	1.5	4.0	1.1	2.2	23.8	5.7	46.8
telnet	5.5	3.1	1.8	0.8	2.7	0.7	4.1	13.1	19.7	48.4

Table 7: Confusion Matrix for the Timing-Based Classifier on GMU Traces (sampling rate = 7)

[6] S. Eddy. Multiple alignment using hidden markov models. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 114–120. AAAI Press, July 1995.

[7] S. Forrest, S. Hofmeyer, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, October 1997.

[8] S. Forrest, S. A. Hofmeyer, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, pages 120–128, May 1996.

[9] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary poisson view of Internet traffic. In *INFOCOM 2004*, March 2004.

[10] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, February 1994.

[11] T. Lane. Hidden markov models for human/computer interface modeling. In *Proceedings of the IJCAI-99 Workshop on Learning about Users*, pages 35–44. International Joint Conferences on Artificial Intelligence, August 1999.

[12] T. Lane and C. Brodley. Sequence matching and learning in anomaly detection for computer security. In *Proceedings of the AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 43–49. AAAI, July 1997.

[13] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissmann. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*. DISCEX 2000, January 2000.

[14] V. Paxson and S. Floyd. Wide-area network traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[15] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[16] A. Schliep, A. Schönhuth, and C. Steinhoff. Using hidden markov models to analyze gene expression time course data. *Bioinformatics*, 19(supplement 1):i255–i263, July 2003.

[17] 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.

[18] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2002.

[19] N. Ye. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, June 2000.

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