Packet Classification in Co-mingled Traffic Streams

Siddharth Maru, Timothy X Brown
Dept. of Electrical, Computer and Energy Engineering
University of Colorado at Boulder, CO 80309 - 0530
{siddharth.maru,timxb}@colorado.edu

***Abstract*—This paper considers the problem of packet classification in a co-mingled traffic stream. Given an encrypted co-mingled stream consisting of different protocol flows originating from different sources; we investigate if it is possible to assign packets to their respective sources and identify the protocol for each source. Encryption makes it difficult to obtain any information from packet headers or payloads. Consequently the only information available to an observer is the packet size, arrival times, direction and power levels. This paper presents a statistical approach that analyses the sizes and power levels of packets belonging to each protocol and uses this information to classify the packets in the co-mingled stream. Results are presented for the classification of a co-mingled stream of upto five different protocols. The results show that it is possible to efficiently classify packets based on sizes, direction and power levels. We see that packets belonging to the HTTP protocol are easiest to classify whereas those belonging to the FTP and IMAP protocols are difficult to separate when co-mingled with each other.**

Keywords: Encryption, co-mingled stream, protocol identification, packet flow separation, packet classification

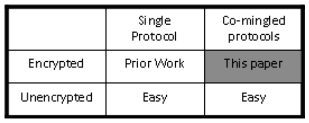
I. Introduction

Flow and packet classification is the estimation of the information content or the identification of the protocol being used in an unknown packet flow. It has been the subject of much research lately [1–6]. Flow and packet classification is important from the point of view of network administrators to identify and block illegal traffic, manage different traffic types and detect intruders. Such classification could also be important from the point of view of an attacker who is trying to jam packets in a wireless network to disrupt communication. If the attacker is able to identify key control packets he can cause significant disruption with minimal jamming effort [7],[8]. For achieving this it is not necessary to correctly identify every packet belonging to the protocol. Even if the attacker is able to classify only a fraction of the control packets, it is sufficient to enable significant disruption. If an attacker is able to classify (separate and identify) packets based on protocol, it amounts to a violation of communication privacy. Also, each individual stream so separated is exposed to further analysis as in [6] which could help an attacker launch further attacks. Hence, such an ability to carry out flow classification is both a privacy as well as a security issue, making its study extremely vital.

We consider an observer of a wired or wireless link in a network. In this paper we focus on a wireless link since it can be observed more easily. The observer is able to capture information about packets on this link and tries to classify them. This is simple if one can inspect packet headers to determine port numbers or inspect packet payloads to determine the protocol to which that packet belongs. However, such methods fail as many applications use dynamic port negotiation or encrypt their packets. Despite encryption, side channel information such as packet sizes and power levels is often available. Thus, it is important to identify what can be inferred from packet sizes, arrival times, packet directions (client to server or vice versa) and packet power levels in the case of a wireless link.

II. Prior Work

Other researchers have studied how to identify the protocol generating an unknown traffic flow. Wright et al. [1–3] have looked at three interesting methods of classification. In [1] they use visual motifs based on packet sizes, arrival times and direction information to identify individual encrypted traffic flows as either FTP, HTTP etc. In [2] they build Hidden Markov Models for each protocol and get a classification accuracy of greater than 75 percent for AOL Internet Messenger, HTTP and HTTPS protocols and less than 50 percent accuracy for SSH and Telnet. In [3] they attempt to identify the application protocol in a stream containing multiple sessions of the same protocol. In [4] Bernaille et al use clustering techniques to identify the application protocol in a single flow. They conclude that they require information regarding the packet sizes of at most the first five packets of each protocol to build a good classifier. Gebski et al. [5] have classified isolated flows represented by request - response packet pairs. They have also looked at identifying the protocols present in streams consisting of two co-mingled protocols. In [6] Sun et al have tried to identify web pages probabilistically based on information such as packet size, arrival times and ordering of packets. As shown in Fig 1, the classification of unencrypted flows is a trivial problem and the classification of a single encrypted flow has been done successfully in the works mentioned above.



	Single Protocol	Co-mingled protocols
Encrypted	Prior Work	This paper
Unencrypted	Easy	Easy

Fig. 1. Classification problems - The shaded box indicates the problem we attempt to solve

Unlike these works, we attempt to classify packets from multiple flows belonging to upto five different protocols (shaded cell in Fig 1). This is a hard problem in general and so we describe several simplifying assumptions. We then test our classifier on some real traffic and present the results of the classification.

III. Packet Classification

A. The Generic Classification Problem

A flow is denoted $a_{ij} \in A$ where $i,\ 0 < i \leq r$ denotes the different sources and $j,\ 0 < j \leq n_i$ denotes the application flows at source i. The set A where $|A| = q = \sum_{i=1}^{r} n_i$, contains all the possible application flows that may be present in the packet stream. A given source can have two flows a_{ij} and $a_{ij'},\ j \neq j'$ which are different instances of the same application, i.e. $a_{ij} = a_{ij'}$. Statistical information, $S(a_{ij})$, regarding the packet sizes, power levels and direction could be computed for each flow. A co-mingled stream is generated by an intermingling of some or all of these flows. Let each flow in a co-mingled stream be denoted as a^*_{ij} and $A^* = \{a^*_{ij}\} \in A$ be the set of all flows actually present in the co-mingled stream, such that $|A^*| = m$.

A co-mingled stream of n packets can be described by the n-vector of data, $D = \{d_1, d_2, ..., d_n\}$ where d_k is the physical features of the kth packet in the stream. The physical features of the packet could differ depending on the situation. In this paper, it consists of the features (size(s), direction(δ)[1], power levels(p)). Let Hypothesis, $H = \{h_1, h_2, .., h_n\}$, where $h_i \in A$ be an assignment of packets to flows. Let $T = \{t_1, t_2, ..., t_n\}$, $t_i \in A$, be the true assignment of packets to application flows. The packet classification problem is *Given D, A, S, compute H such that $P(h_i \neq t_i)$ is minimized. The probability distribution is over different realizations of D and T.*

[1]Internet download traffic (server to client) is generally greater than upload traffic (client to server) which would help us determine which packets are moving from server to client and which the other way.

B. Two Approaches to the Classification problem

The generic problem could be solved in two ways: Classifying the co-mingled stream by first assigning the packets to sources and then assigning packets belonging to each source to their respective application protocols. The other way is to classify the co-mingled stream into flows directly using D and statistical information S. This latter joint approach is used in this paper.

C. Why is the Generic Classification Problem hard?

The generic problem could be tackled at various levels of difficulty. One might have access to physical layer information such as flow identifiers as in the case of some wireless protocols making it very easy to separate the stream based on sources [8]. These substreams now only need to be classified into their respective applications. On the other hand, we might not have any distinguishing information for traffic from various sources. The same source can send more than one traffic stream belonging to the same application type. Different sources may be sending traffic from the same application type. Thus, assigning packets to an application or assigning packets to a source are not enough to assign packets to a specific flow. Since a protocol can access information from anywhere in the world, the round trip times vary significantly, making timing information unreliable. The intermingling of packets makes it difficult to look for a certain sequence of packet sizes as successive packets could belong to different flows. Packet power levels can vary with time and distance from the source. Hence, the power levels of packets being transmitted by the same source exhibit a distribution of values instead of a distinguishing constant value. Data and ACK packets for busy TCP flows are identical for any application. Control packet sizes of different protocols can be identical. This is aggravated by encryption which can pad packets to the next encryption block size. For instance [5] rounds up packets to the next 16 byte boundary[2].

D. Simplified Classification problem

We make certain assumptions to simplify this problem.

1) We assume that $A = A^*$ and consequently $m = q$, i.e. we know exactly what protocols are present in the co-mingled stream. But, we do not know which packets belong to which protocol or which stream.
2) $n_i = 1$ i.e. each source transmits using only one application protocol.
3) No two sources transmit using the same protocol.
4) We assume we can collect the statistics $S(a_{ij})$ for

[2]This is the case with FTP and IMAP protocols many of whose control packet sizes are similar (caused by round off due to encryption) making it difficult to differentiate between them based on sizes.

each flow $a_{ij} \in A^*$. The statistics and how they are collected is described in sections III and IV.

In addition to these simplifying assumptions, we make two assumptions that make the problem realistic. The first is that one of the sources is the residual background traffic that is always present on the links whether there is a specific user level application or not. The second is that encryption causes rounding of the packet sizes to the nearest 16 byte boundary. Effectively, we have statistical information about the packet sizes and power levels of all the application protocols and background traffic present in the co-mingled stream.

IV. Solving the classification problem

One approach to this problem is the Maximum Likelihood approach. The aim is to determine using S, A and D the hypothesis with the maximum likelihood, given by $H_{ML} = \arg\max_H P(H|D)$. Thus, the likelihood of each of the hypothesis needs to be computed and the one with the maximum value picked as the correct classification. This is theoretically the best that one could do in this classification problem.

The maximum likelihood solution can be re-written as $\arg\max_H P(h_1, h_2, ..., h_n|D, A, S)$. Each stream of length n could be split into m substreams such that each substream contains all the packets from a particular protocol. We assume that these substreams are independent of each other and hence their likelihoods could be computed separately and the results multiplied to give us the overall likelihood. We analyze one such substream here. We denote the hypothesis for the ith substream containing n^i packets as $H^i = \{h^i_1, h^i_2, ..., h^i_{n_i}\}$ and corresponding data as $D^i = \{\vec{s^i}, \vec{p^i}, \vec{\delta^i}\}$, where $\vec{s^i}$, $\vec{p^i}$ and $\vec{\delta^i}$ are the vectors of size, power levels, and directions of the packets in the substream. The likelihood can then be written as

$$P(H^i|D^i) = P(h^i_1, h^i_2, ..., h^i_{n^i}|\vec{s^i}, \vec{p^i}, \vec{\delta^i})$$

To simplify notation, we implicity assume the conditioning on the statistics(S). Simplifying, we get

$$P(h^i_1, h^i_2, ..., h^i_{n^i}|D^i) = P(h^i_1|D^i) \cdot P(h^i_2, h^i_3, ..., h^i_n|h^i_1, D^i)$$
$$= P(h^i_1|D^i) \cdot P(h_2|h^i_1, D^i) \cdot P(h^i_3, ..., h^i_n|h^i_1, h^i_2, D^i)$$

However, we assume that any packet is dependent only on the packet immediately preceeding it (and not the ones further in the past). This is appropriate as we are only considering pairwise statistics in our classifier. Hence, continuing the previous equation:

$$= P(h^i_1|D^i) \cdot P(h^i_2|h^i_1, D^i) \cdot P(h^i_3, ..., h^i_n|h^i_2, D^i)$$

$$= P(h^i_1|D^i) \cdot P(h^i_2|h^i_1, D^i) \cdot ... \cdot P(h^i_{n^i}|h^i_{n^i-1}, D^i) \quad (1)$$

Thus, we need to compute $P(h^i_1|D^i)$ (discussed in IV-A) for the first packet and $P(h^i_j|h^i_{j-1}, D^i)$ for each subsequent pair of packets in the substream to compute the likelihood for the substream.

We now simplify the second term $P(h^i_2|h^i_1, D^i)$.

$$P(h^i_2|h^i_1, D^i) = P(h^i_2, h^i_1, D^i)/P(h^i_1, D^i)$$
$$= P(h^i_2, h^i_1, s^i_1, s^i_2, p^i_1, p^i_2, \delta^i_1, \delta^i_2)/P(h^i_1, s^i_1, s^i_2, p^i_1, p^i_2, \delta^i_1, \delta^i_2)$$

Let N be the numerator and Y the denominator of the above equation. Assuming power to be independent of packet size and $h^i_1 = h^i_2 = h^i$ (since every packet in a substream belongs to the same protocol); we get,

$$N = P(p^i_1|\delta^i_1, h^i)\ P(p^i_2|\delta^i_2, h^i)\ P(s^i_2, \delta^i_2|s^i_1, \delta^i_1, h^i)$$
$$P(s^i_1, \delta^i_1|h^i)\ P(h^i)$$

$$Y = P(p^i_1|\delta^i_1, h^i)\ P(s^i_1, \delta^i_1|h^i)\ P(h^i)\ P(s^i_2, \delta^i_2|s^i_1, \delta^i_1)\ P(p^i_2|\delta^i_2)$$

$$N/Y = P(h^i|D^i)$$
$$= P(s^i_2, \delta^i_2|s^i_1, \delta^i_1, h^i) \cdot P(p^i_2|\delta^i_2, h^i) \cdot B \quad (2)$$

where, $B = 1/(P(s^i_2, \delta^i_2|s^i_1, \delta^i_1) \cdot P(p^i_2|\delta^i_2))$
The factor B is independent of the protocol and hence common across all substreams. To compute each of the pairwise terms in equation (1), we need to compute equation (2). The factors on the right in equation (2) apart from B constitute the pairwise probability of the packet pair (discussed in section IV).

If we consider all possible assignments of packets to flows, there are q^n different hypotheses and a brute force approach at solving this problem would consume exponential amount of time.

Stream Classification consists of four stages as shown in Fig 2.

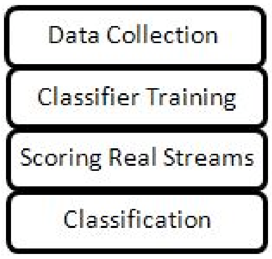


Fig. 2. The figure shows the four stages in the classifier

Data Collection: Traffic traces for various TCP-based application protocols are collected over a communication link using Wireshark [9]. The key control packets for different protocols have varying sizes and hence are of interest to us. Consequently, we retain all packets of size less than 1000 bytes from the traffic traces and the others are removed[3]. Enough data should be collected for each protocol to make an accurate representation of the packet size and power level distributions during the training stage.

Classifier Training: The data sets collected during the data collection stage are used to build a priori probability and pairwise probability tables (discussed later). These tables are used in the next stage.

[3]The data packets (large packets) across protocols are likely to be of the maximum MTU size and hence identical from our point of view. Also, most of these larger packets are lumped together and hence do not exhibit a recurrent pairwise pattern as do the smaller packets.

Generation of Score Matrix: The score matrix (X) is a data structure generated for each test traffic stream that we wish to classify. It is an efficient means of storing the information from the a priori and pairwise probability tables. It is not a new statistic.

Classification: In the classification stage, a new unknown stream is analysed with the help of the score matrix to carry out classification.

V. Algorithm Data Structures

We look at three critical tables necessary for carrying out classification. These are estimated from training sequences of packets from a known protocol.

A. A priori probabilities tables

One table is generated for each protocol involved. For each packet in the training data, we estimate $\hat{P}(h_i|s_i, p_i, \delta_i)$, where s_i, p_i and δ_i are the size, power and direction of the packet respectively i.e. the probability that the packet belongs to some protocol given the size, direction and power level of the packet.

$$\hat{P}(h_i|s_i, p_i, \delta_i) = P(s_i, p_i, \delta_i|h_i) \cdot P(h_i)/C$$

where,

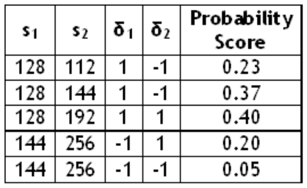
$$C = \sum_{k=1}^{m} P(s_k, p_k, \delta_k|h_k) \cdot P(h_k)$$

Hence, the a priori probability based on size, direction and power level is

$$\begin{aligned}\hat{P}(h_i|s_i, p_i, \delta_i) &= P(p_i|s_i, \delta_i, h_i) \cdot P(s_i, \delta_i|h_i) \cdot P(h_i)/C \\ &= P(p_i|\delta_i, h_i) \cdot P(s_i, \delta_i|h_i) \cdot P(h_i)/C\end{aligned}$$

Since C is constant for all protocols, it is ignored. In the absence of power level information, the factor $P(p_i|\delta_i, h_i)$ defaults to 1.

B. Pairwise probabilities tables



s_1	s_2	δ_1	δ_2	Probability Score
128	112	1	-1	0.23
128	144	1	-1	0.37
128	192	1	1	0.40
144	256	-1	1	0.20
144	256	-1	-1	0.05

Fig. 3. Pairwise prob. table

One pairwise probability table (Fig 3) is created for each protocol. The tables contain entries for each pair of successive packets in the training data. Consider a pair of packets x_1, x_2 having sizes s_1, s_2, directions δ_1, δ_2 and power levels p_1, p_2 belonging to protocol k. We estimate the pairwise probability $\hat{P}_{x_1,x_2} = P(s_2, \delta_2, p_2|s_1, \delta_1, p_1, h_1 = k, h_2 = k) \cdot P(h_2)$ $= P(p_2|\delta_2, h_2) \cdot P(s_2, \delta_2|s_1, \delta_1, h_1, h_2) \cdot P(h_2)$, assuming power is independent of packet sizes. This is the pairwise probability based on packet sizes, direction and power levels. In the absence of power level information (as in Fig 3), the factor $P(p_2|\delta_2, h_2)$ defaults to 1.

C. Score Matrix

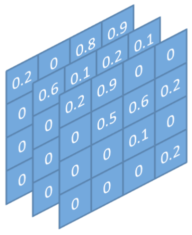


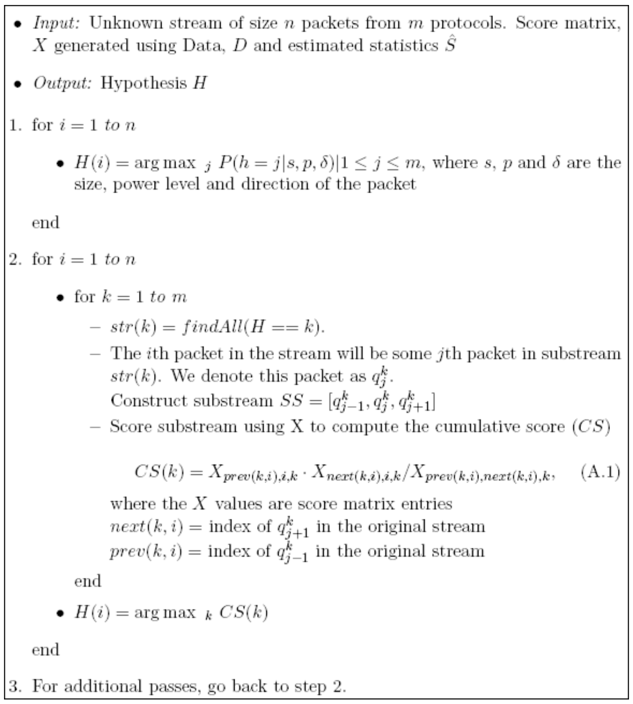
Fig. 4. Score matrix

The score matrix can be generated during either the training stage or at run time. If generated during the training stage the matrix would contain every possible combination of packet sizes and directions. Hence, to reduce the run time memory requirement, the matrix is generated at run time. For a test stream of length n packets containing m co-mingled protocols, a score matrix of dimensions $n \times n \times m$ is generated. Each pair of successive packets in the test stream is searched in the pair wise probability tables and the corresponding probability score is populated in the score matrix at the intersection of the row and column signifying that packet pair (Fig 4). Hence,

$$X_{i,j,k} = \begin{cases} P(s_i, \delta_i|h_i) \cdot P(p_i|\delta_i, h_i) \cdot P(h_i) & \text{if } i = j, \\ P(s_j, \delta_j|s_i, \delta_i, h_i, h_j) \cdot P(p_j|\delta_j, h_j) \cdot P(h_j) & \text{if } i \neq j, \\ 0 & \text{if } i < j. \end{cases}$$

where $1 \leq k \leq m$. If the score matrix is sparse, it could be more efficiently stored in hash tables. In either case, the access time for any entry in the score matrix has time complexity $O(1)$.

VI. Classification Algorithm



```
• Input: Unknown stream of size $n$ packets from $m$ protocols. Score matrix,
  $X$ generated using Data, $D$ and estimated statistics $\hat{S}$

• Output: Hypothesis $H$

1. for $i = 1$ to $n$

      • $H(i) = \arg\max_j P(h = j|s, p, \delta)|1 \leq j \leq m$, where $s$, $p$ and $\delta$ are the
        size, power level and direction of the packet

   end

2. for $i = 1$ to $n$

      • for $k = 1$ to $m$
          – $str(k) = findAll(H == k)$.
          – The $i$th packet in the stream will be some $j$th packet in substream
            $str(k)$. We denote this packet as $q_j^k$.
            Construct substream $SS = [q_{j-1}^k, q_j^k, q_{j+1}^k]$
          – Score substream using X to compute the cumulative score $(CS)$

                $CS(k) = X_{prev(k,i),i,k} \cdot X_{next(k,i),i,k}/X_{prev(k,i),next(k,i),k},$     (A.1)

            where the $X$ values are score matrix entries
            $next(k, i) =$ index of $q_{j+1}^k$ in the original stream
            $prev(k, i) =$ index of $q_{j-1}^k$ in the original stream
        end

      • $H(i) = \arg\max_k CS(k)$

   end

3. For additional passes, go back to step 2.
```

Fig. 5. Classification Algorithm

For an unknown stream the score matrix is generated. The complete stream could be scored using equation (1).

However, the aim is to locally determine the best assignment for a given packet using statistical information regarding that packet and its immediate neighbors as in Fig 5. The numerator in (A.1) is the score if protocol k is assigned to packet i and the denominator is the score when protocok k is not assigned to packet i. If this ratio is greater than one for any protocol k, it would mean that the score would be higher if that protocol is assigned to this packet (i) rather than not. If the current packet is the first packet for a protocol, $X_{prev(k,i),i,k}$ defaults to $X_{i,i,k}$ which is the diagonal entry of the score matrix and $X_{prev(k,i),next(k,i),k}$ defaults to $X_{i,next(k,i),k}$. If the current packet is the last packet for a protocol, $X_{i,next(k,i),k}$ and $X_{prev(k,i),next(k,i),k}$ default to 1 as there is no packet following it.

Assuming that the score matrix is generated during the training phase using hash tables, the time complexity of step 1 of the algorithm is $O(nm)$, since the maximum is being computed over m protocols. The second step which is an additional pass over the classification obtained from the first step consists of nested *for* loops and hence is also $O(nm)$. Hence, the overall complexity of this approach is $O(nm)$.

VII. Experiments and Results

Protocol	Distribution
HTTP	N(3,1)
Torrent	N(4,1)
FTP	N(5,1)
IMAP	N(6,1)
Back	Uniform[2,7]

Fig. 6. Assumed power level distributions for the protocols

We attempt to classify some commonly used protocols using our classifier. The traffic was generated by running an instance of an application for each of HTTP, Torrent(BT), FTP and IMAP protocols as well as residual background traffic (Back). Packets from each protocol were captured on a live unencrypted link using Wireshark[9]. Distinct power level distributions were then assigned to the packets of each of the application protocols to generate a trace containing both packet size and power level information. Assuming that the power levels are measured in dB, the measured power levels are distributed log normally[10]. Hence, the four TCP-based protocols were assumed to have Gaussian distributed power levels whereas background traffic was assumed to have a uniform distribution spread over the power levels of the four protocols (Fig 6). Assuming that we are observing and collecting traffic traces at some access point, the power level distributions apply to only the traffic being sent to the access point as all the packets being sent the other way have the same power. Hence, we assume power level information in only one direction. We assume that all protocols are equally likely ($P(h_i)$ is the same for all protocols).

All the packets of size greater than 1000 bytes (presumed data packets) were discarded from the traces. Since the SYN and ACK packets would be present for every protocol (and hence difficult to classify), they too were discarded[4].

	Total	HTTP	BT	FTP	IMAP	Back
HTTP	33	27	5	0	1	0
BT	42	2	29	4	4	3
FTP	50	2	13	22	13	0
IMAP	48	2	8	5	26	7
Back	46	0	4	0	2	40

Fig. 7. Confusion Matrix - The totals on the left indicate the actual number of packets in each protocol (219 total). The numbers under the vertical columns show to which protocol each was classified

There were 8000 to 15000 control packets in each of the traces. These traces were input to the training stage and used to build the score matrix. Test traces containing different number and different combinations of the four protocols interspersed with background traffic were collected. The test traces were generated in such a manner as to ensure that each of them had 195 to 240 control packets total. These traces were then classified and the results expressed in terms of confusion matrices. Next, we discuss some of our results. For the Results 1 and 2, only the knowledge of packet sizes was assumed whereas for Result 3, knowledge of both packet sizes and power levels was assumed.

A. Result 1: Classification Performance

The classification algorithm was run on five test sets with five co-mingled protocols assuming the knowledge of only packet size distributions and the classification results expressed in terms of confusion matrices. The confusion matrices over five runs were averaged to give the matrix shown in Fig 7. It can be seen that the packets belonging to the HTTP protocol were the easiest to classify while the most mistakes were made in classifying packets belonging to the FTP protocol.

B. Result 2: Two protocol analysis

The algorithm was run on six datasets each containing a different combination of two of the four protocols in background traffic and the classification results expressed as confusion matrices (Fig 8). The matrix in Fig 9 which was generated from Fig 8 shows how the protocols are confused with each other. We see that in a co-mingled stream containing HTTP and BT protocols in background traffic, 87% of HTTP control packets were correctly identified, while only 70% of BT packets were correctly identified. 11% of HTTP control packets got classified as BT packets and 14% of BT packets got

[4]Though some FTP control packets may be confused and hence discarded along with SYN and ACK packets due to similarity in size, we accept the loss of a few such packets for an increase in the computational efficiency obtained through discarding a large number of SYN and ACK packets.

	Total	HTTP	BT	Back
HTTP	61	53	7	1
BT	50	7	35	8
Back	84	0	6	78

	Total	BT	FTP	Back
BT	86	68	12	6
FTP	80	14	64	2
Back	74	4	0	70

	Total	HTTP	FTP	Back
HTTP	41	37	4	0
FTP	86	5	74	7
Back	84	0	5	79

	Total	BT	IMAP	Back
BT	89	69	10	10
IMAP	77	18	49	10
Back	74	4	0	70

	Total	HTTP	IMAP	Back
HTTP	46	41	5	0
IMAP	74	6	55	13
Back	89	2	10	77

	Total	FTP	IMAP	Back
FTP	81	47	31	3
IMAP	86	13	55	18
Back	73	6	4	63

Fig. 8. Pairwise classification performance between protocols

Protocol		Classification Accuracy			
A	B	% A as A	% B as B	% A as B	% B as A
HTTP	BT	86.9	70.0	11.5	14.0
HTTP	FTP	90.2	86.1	9.8	5.8
HTTP	IMAP	89.1	74.3	10.9	8.1
BT	FTP	79.1	80.0	13.9	17.5
BT	IMAP	77.5	63.6	11.2	23.4
FTP	IMAP	58.0	63.9	38.3	15.1

Fig. 9. Mixing Matrix for the protocols

classified as HTTP packets and the rest got classified as background traffic. If both protocols have high classification efficiencies and low mixing percentage, then they are considered easy to separate. As per this yardstick, the BT-FTP combination and the FTP-IMAP combination are the hardest to separate. This is due to the similarity in size of the control packets in the two protocols. The HTTP and FTP protocols are the easiest to separate.

C. Result 3: Impact of inclusion of power levels

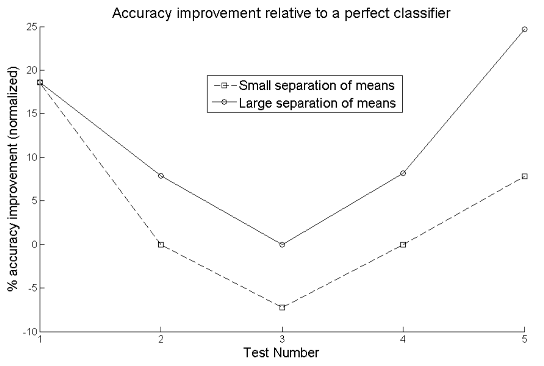


Fig. 10. Impact of including power level information

Tests were carried out on the five datasets used for Result 1. Given prior information about the power level distributions (Fig 6), the classification accuracies improved in comparison to classification in absence of power level information (Fig 10). The separation in the means of the power levels was doubled to show how greater separation improves classification accuracy (solid curve).

VIII. Conclusions and Future Work

It is possible to identify the key control packets using packet size, direction and power level information. The HTTP-FTP protocol combination is easy to separate whereas the separation of the FTP-IMAP combination is difficult. Also, the inclusion of power level information leads to improved classification accuracies. This work made several simplifying assumptions. Future work will seek to relax these assumptions, specifically exploring classification performance when $m < q$ and allowing values of n_i greater than one. For the case when $m < q$, it would be necessary to estimate the correct value of m to prevent a decrease in classification accuracy. For example, the number of modes in the aggregate power level distribution could be estimated. When $n_i > 1$, further analysis would be required.

Acknowledgements

This work has been supported by the FAA.

References

[1] Charles V. Wright, Fabian Monrose, and Gerald M. Masson. Using visual motifs to classify encrypted traffic. In *VizSEC '06: Proc. of the 3rd international workshop on Visualization for computer security*, pages 41–50, New York, NY, USA, 2006. ACM.

[2] Charles Wright, Fabian Monrose, and Gerald M. Masson. HMM profiles for network traffic classification. In *VizSEC/DMSEC '04: Proc. of the 2004 ACM workshop on Visualization and data mining for computer sec.*, pages 9–15, New York, NY, USA, 2004. ACM.

[3] Charles V. Wright, Fabian Monrose, and Gerald M. Masson. On inferring application protocol behaviors in encrypted network traffic. *J. Mach. Learn. Res.*, 7:2745–2769, 2006.

[4] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, April 2006.

[5] Matthew Gebski, Alex Penev, and Raymond K. Wong. Protocol identification of encrypted network traffic. In *WI '06: Proc. of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 957–960, Washington, DC, USA, 2006.

[6] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy*. Society Press, 2002.

[7] Timothy X Brown, Jesse E. James, and Amita Sethi. Jamming and sensing of encrypted wireless ad hoc networks. In *In Proc. of MobiHoc '06*, pages 120–130. ACM, 2006.

[8] Siddharth Maru and Timothy X Brown. Denial of service vulnerabilities in the 802.16 protocol. In *WICON '08: Proc. of Conf. on Wireless Internet*, pages 1–9, ICST, Brussels, Belgium, 2008. ICST.

[9] CACE Technologies. Wireshark. `http://www.wireshark.org/about.html`.

[10] Theodore S. Rappaport. *Wireless Communications: Principles and Practice (2nd Ed.)*. Prentice Hall, Dec. 2001.