Network Abuse Detection via Flow Content Characterization

Mehdi Kharrazi, Kulesh Shanmugasundaram, Nasir Memon

Abstract— **One of the growing problems in our networks is the abuse of computing resources by authorized and unauthorized personnel. The nature of abuse may vary from using unauthorized applications to serving unauthorized content. Proliferation of peer-to-peer networks and the availability of of proxies for tunneling makes it difficult to detect such abuse and easy to circumvent security policies. This paper presents a novel method that can be used to detect abuse of resources on a network based solely on the payload content type. The proposed method does not depend on packet headers and other simple packet characteristics and hence is able to better detect incidents of abuse.**

I. Introduction

Abuse of computing resources is one of the growing problems for which there hasn't been a feasible solution suggested yet. Network administrators routinely have to deal with abuse of network resources such as network bandwidth by unauthorized application services, and the distribution of unauthorized content to name a few. Abusers can be malicious attackers looking for free resources to host their illegal activities, a malicious insider running a peer-to-peer hub, or simply an ill informed user unintentionally running an application proxy.

The two most common defenses that are used to prevent network abuses are Firewalls and Intrusion Detection Systems (IDS). An IDS is not useful in detecting many types of abuses because the essence of the abuse is not captured by a simple set of signatures. Firewalls, on the other hand, are somewhat effective in preventing abuse. Currently, firewalls use port blocking to thwart unauthorized application services. By convention [1], certain port numbers are bound to certain types of applications (or application level protocols) and are referred to as "well-known ports." For instance, port 80 is HTTP and port 22 is SSH. Therefore, if a security policy denies the use of web servers inside a network then a firewall simply blocks traffic to port 80.

However, it is now well known that a firewall can be circumvented under certain circumstances. For example, most firewalls do not block outbound connection requests. A malicious insider or a host inside the network compromised by an attacker can initiate a connection and transfer unauthorized data or make available an unauthorized service without being detected by a firewall. Another simple way to by pass the firewall would be to simply run the unauthorized service on a port that the firewall allows traffic on. So for example, if the firewall blocks services on port 80 and leaves port 22 open so that users can telecommute, then a web server can be configured to use port 22, thereby circumventing the security policy. A third way to get past the firewall is by tunneling. Tunneling is a technique that exploits some protocol-family's ability to move packets from user to user, or to open virtual-circuits between users, and use this as if it were a data-link protocol to run another protocol family's upper layers (or even the same protocol family's upper layers). Tunneling works by encapsulating a network protocol within packets carried by another protocol. So in the above example, with the presence of a suitable proxy on the inside host, web traffic could be tunneled through SSH traffic on port 80. Similarly, there are many other techniques that hackers employ to get past a firewall, given a malicious insider or a captured host inside the target network.

ISIS Laboratory, Department of Computer & Information Science, Polytechnic University, Brooklyn, New York, NY 11201

Firewall circumvention techniques give rise to new challenges in abuse detection. In this paper we present a method to detect network abuse based on *Flow Content Characterization.* By flow content characterization we mean the ability to classify network packet contents as belonging one of a set of data types like audio data, encrypted data, video data etc. Note that our intention is not to identify the application being used but to identify the type of content emanating from a host or a network flow in general. Flow content characterization can be very useful in abuse detection. The knowledge of content type emanating from a host can help detect abuses of resources on a network. For example, suppose we have a FTP server that serves RFCs to the Internet. Using flow content characterization if we found that the server is emanating encrypted-text or multimedia content then we know that the server is being used for purposes other than serving RFCs. As another example, observe that in general, a particular port has certain flow content characteristics. For instance, port 80 generally carries a lot of plain-text, some compressed-text, and multimedia content. Knowing this flow content characteristic for a network, we can monitor for significant deviations in the characteristics for detecting the presence of tunneling. If we observe that 80% of traffic on port 80 is audio traffic then it may indicate that some one is tunneling audio traffic through port 80. Similarly, tunneling various applications through port 80 will show significantly different characteristic than normal web traffic.

One critical question arises, however. Why can't we simply use unique header information found in multimedia formats to identify the content type? First of all, media headers (like the JPEG headers or MPEG headers) can be modified easily. Therefore, it is easy to circumvent a method that relies on the header information. On the other hand, not every single packet contains header information. For instance, suppose there is a 200KB JPEG image. When transmitted over network this image will be split into 200 packets and only one of them contains a header. A header based monitoring system must be able to examine each packet on the network for the string "JFIF" to determine the content type is a JPEG image. This is obviously a very expensive process in terms of processing power and memory and such an approach is not viable on a large network where the traffic volume is high. Besides, packet drops may result in this method losing the packet that has the header information rendering it useless. Also note that some media types do not have headers at all. For example, plain-text and encrypted content usually have no headers to indicate their content type. Hence, the proposed method does not rely on media headers. The method samples packets from a network, groups them into flows and uses the group of packets to characterize the flow content based on statistical properties.

The rest of the paper is organized as follows: in the following section we briefly present the related work. Section III presents various statistical measured used in characterization followed by a description of experiments and results in Section IV. We present a discussion of the results in Section V. We conclude with a summary of our current accomplishments and future work in Section VI.

II. Related Work

Over the past few years significant research has been done to characterize network flows. Network traffic characteristics of various applications such as, web, email, and multimedia streaming, have been studied to support emerging network traffic trends for improving the underlying protocols. For this purpose, researchers have looked into various characteristics of network traffic such as, size of packets, inter-packet timings, round trip times, and transmission protocols. Network security community have borrowed some of these ideas and extended some others to improve the security of networks by identifying malicious network flows, applications, or hosts. In this section we briefly discuss prior work on network traffic characterization related to network security and refer the readers to [4] for a survey on traffic characterization in general.

In [2] a neural network is used to detect the presence of unauthorized network services. The author proposes a method that uses the neural network to learn the signature of common network services and then monitor the network to detect flows that deviate from the norm. The authors used the total number of bytes transferred as a single feature to distinguish between Telnet and FTP traffic on networks. In [3], authors propose a method to identify well-known applications being tunneled through unconventional ports. The proposed method uses a decision tree algorithm to learn the statistical properties of various applications. The model learned is then used to characterize the application types of network flows. Thanks to weak port bindings, port numbers are not considered in learning the model of both of these works.

A related problem to the above works is tracing connection chains over multiple networks. Attackers often obscure their identity and location by forming a connection chain by logging into a set of compromised systems before attacking a target. Tracing the attack from the victim takes us only to the last link in the chain but not to the location of the attacker. In [5], [6], methods are proposed to trace intruders who obscure their identity by logging through a chain of multiple machines, known as stepping-stones. The method proposed in [5] creates "thumb-prints" of connections using packet content which can be compared to determine whether two connections contain the same content and are therefore likely to be part of the same connection chain. However, the method fails when the connections are encrypted. To address the problem [6] proposes an algorithm that doesn't rely on traffic content, instead relies on packet sizes, packet intervals, etc. to detect stepping stones.

III. Identifying Measures

In order to distinguish between a variety of flow content types, we look at the payload in each packet as a 1-D vector of 8-bit numbers. Thus, our goal is to come up with a statistical model that would help us distinguish these vectors based on their respective statistical signatures. The statistical measures we considered to build the model can be separated into three broad categories: time domain, frequency domain, and higher order statistical measures. We review these measures in the rest of this section and give intuitive reasoning for their selection.

- *Time Domain*: We choose a number of simple statistical measures from the time domain. Although some of these measures are simple and rudimentary, they help greatly in distinguishing different content types. These measures are, mean, variance, auto-correlation, and entropy. For example, since we are considering the payload of captured packets as 8-bit integers, the mean value of each fragment should lie anywhere between 0 and 255. Now, one would expect compressed or encrypted data to have a mean around 128, where as flows carrying a plain-text file should have lower mean values. The expectation is reasonable because compression and encryption de-correlates data or randomize it thereby distributing the values uniformly whereas in

plain-text format, where ASCII standard is used, most of the symbols in printable text have a numerical representation between 32 to 127 and the extended ASCII codes which range from 128 to 255 are used rarely, thus lowering the mean value.
Another measure we consider from the time domain is entropy. Entropy, $H(x)$, is defined as $H(x) = -\sum_x P(x)log_2(x)$ and is a measure of randomness in the data. One would expect that RAW data formats such as, bitmaps images, or .WAV audio, to have lower entropy than compressed or encrypted formats. This is evident in figure 1 which shows the average entropy of data fragments for 1000 files in each of the 8 categories. Discussion on how the data set used was obtained is in Section IV. Similar reasoning can be used for using variance and auto-correlation as well.

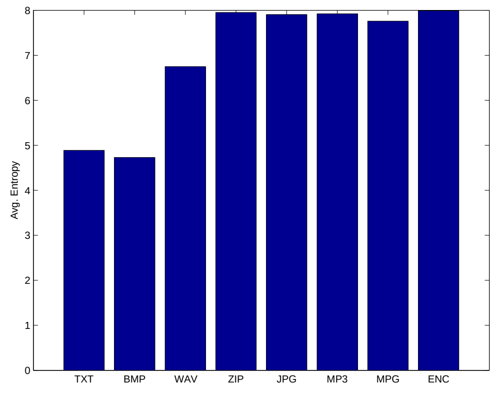

Fig. 1. Average entropy of data vectors from 8 different file types.

• *Frequency Domain*: As stated in the previous paragraphs it is expected that payload data originating from different file types should have different statistical properties. This led us to inspect the frequency domain representation of a set of data vectors obtained from different files. Seeing that on average there are subtle difference in their frequency representations depending on the original data type, we choose to use a number of statistics from the power spectrum. To that order we first divide the power spectrum into 4 separate bands ranging from, $0-\pi/8$, $\pi/8-\pi/4$, $\pi/4-\pi/2$, and $\pi/2-\pi$. We then calculate the mean, variance and skewness of each band. For example the average mean of the power spectrum in the first band could be seen in figure 2.
• *Higher Order Statistics*: The last category of measures which we use are higher order statistical measures. More specifically we look at bicoherence, which is a third order statistic. The bicoherence is able to characterize nonlinearities in the underlying data. Our argument is that the amount of non-linearity introduced by the compression or encryption techniques varies. Thus these measurers could help in separating these data types. We first calculate the

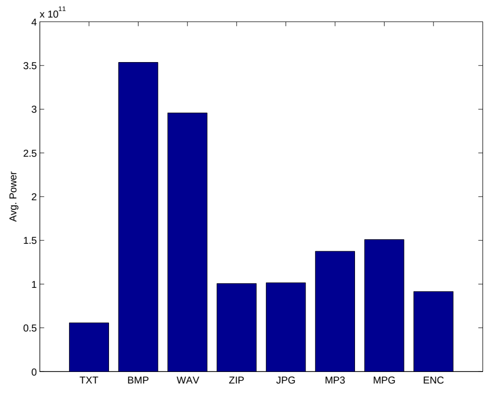

Fig. 2. Average mean of the first band, $0-\pi/8$, of the power spectrum for data vectors from 8 different file types.

bicoherence, after which the following statistics are calculated, power of the bicoherence magnitude and phase, and also the mean of the bicoherence magnitude. In addition to these statistics we also calculate the kurtosis, and skewness of each data vector. For a good review of bicoherence and more generally on higher order statistics the reader is referred to [16]. For example, skewness is a measure of symmetry of a given distribution, if the negative tail is larger than the positive tail, then we will have a negative skew, and if the positive tail is larger than the negative tail we will have a positive skew. The average skewness for the 8 content types are shown in Figure 3. It is clear that depending on the content type the average skewness varies greatly. Furthermore, in the case of encrypted data, the average skewness is zero meaning that it should have a normal distribution.

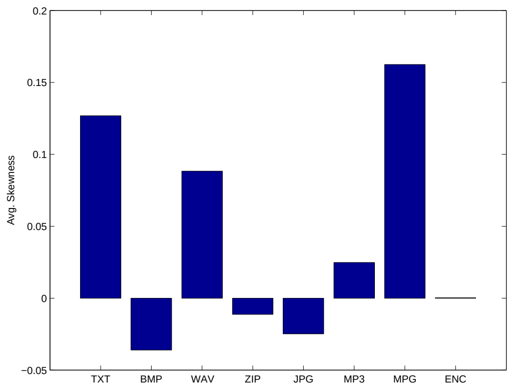

Fig. 3. Average skewness of data vectors from 8 different file types.

We need multiple statistics to distinguish between the different data types as the differences between them are complex and cannot be easily captured by a single statistic. For example, observing Figure 1 we can see that entropy can be used to distinguish between {TXT, BMP} and rest of the content types. It cannot be used to distinguish between TXT and BMP. On the other hand, power spectrum (See Figure 2) can be used to distinguish between BMP and rest of the content types. Combining both entropy and power spectrum together we can distinguish between TXT and BMP. Similar examples can be provided for the other data types.

IV. Experiments

In this section we describe the experiments carried out to evaluate the effectiveness of our method in classifying the flow content. We describe our data set, the experimental setup, and the results.

A. Data Set

There are a variety of content types available on the Internet. One could divide these content types into three major categories: raw (or uncompressed), compressed, and encrypted data. Our goal is to evaluate how well we can distinguish between data from each category. To this end, we have selected a number of different content types from each category. For example, in the raw category we look at content types of plain-text, BMP, and WAV, and in the compressed category ZIP, JPG, MP3, and MPEG files. Our data set, consisting of the 7 different file types are obtained from a random crawl of a peer-to-peer network. The only constraint placed on the downloads was that the files be at least 50KB. This is to ensure that that maximum length of data vectors in the experiments can be at least 50KB. A total of 1000 files are downloaded for each file type. These files are then encrypted using the AES encryption algorithm to obtain 1000 encrypted files.

B. Classification

There are a variety of classification algorithms available and we have chosen to use Support Vector Machines [17] in our experiments. The selection is based on our previous experiments on a number of different data sets, and observing better performance results from SVM as compared to other classifiers. In essence, there are two main differences between a SVM classifier and a linear classifier. In SVM:

- The data is mapped to another dimension, this is done with the help of kernels. There a multitude of kernels to choose from such as Polynomial, Radial, sigmoid.
- A separating hyperplane is chosen so that the separating margins are maximized.

In our experiments we opted to use the RBF kernel (Radial Basis Function). The RBF kernel is optimized by doing a grid search over it's two parameters, cost and gamma. There are many implementations of SVM available on the public domain. We have chosen the freely available LibSvm [18] package.

C. Experimental Setup

The proposed statistics are computed over various sizes of payload. In order to simulate sampling packets off the wire we segmented each file into 1024 byte blocks. The 1024-byte block is chosen in accordance with the size of average TCP packet size. (See Figure 5). 32768 bytes of data, or equivalently 32 packets are collected from random locations in each file. Since we are interested in seeing the effects of the size of available data on the classification results, we then obtained differen size payloads from the 32768 bytes of sampled data. Given a payload size, statistics are then calculated for the payload after which we have 1000 feature vectors, containing the identifying features proposed, for each of the eight categories. A SVM classifier is then trained using 400 feature vectors from each of the 8 categories. The remaining 600 features vectors are then used to test the resulting classifier.

D. Results

The above procedure is repeated for different payload sizes, and as expected the accuracy of the classification improves with the size of payload considered for classification. In our experiment accuracy is defined as:

$$Accuracy = \frac{T}{T+F} \tag{1}$$

where T is number of samples classified correctly, and F is the number of samples classified incorrectly. Figure 4 shows the results of accuracy vs. payload size tradeoff. Interestingly, however, we observe that the accuracy begins to saturate as the features are computed over payloads larger than $16KB$.

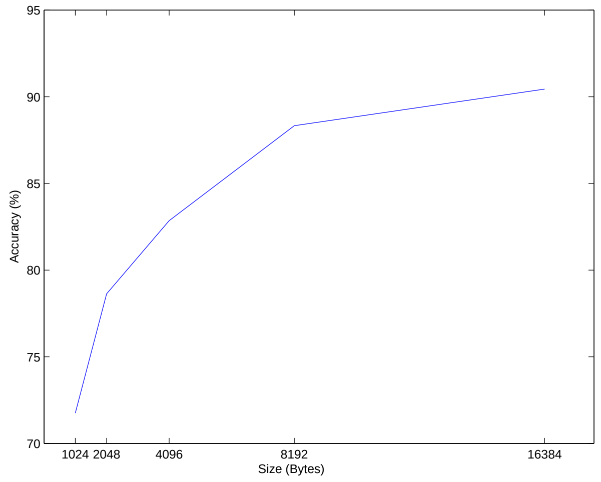

Fig. 4. Payload size vs. accuracy.

Since we are building a multi-class classifier, just looking at the overall accuracy would not give a complete picture on how well the classifier is able to distinguish between the categories. In order to see how well each category is distinguished with respect to the other, we compute the confusion matrix. Confusion matrix presents information about the actual and predicted results using the classifier. These entries should not be misinterpreted as accuracy figures. In fact, the overall accuracy of the classifier is equal to the average of the diagonal entries in the confusion matrix. Table I presents the confusion matrix for payload of 1024 bytes. For example, from the table we can see that 97.5% of plain-text payloads are correctly classified. However, 1.83%, 0.33%, 0.17%, and 0.17% of them are incorrectly classified as BMP, WAV, ZIP, and MPG respectively. On the other hand, the detection rate for the ZIP format is 33%, and it is predicted incorrectly as encrypted format 37.5% of the times.

As shown in figure 4, the overall accuracy of classification improves when we consider larger payload chunks. Thus, we also compute the confusion matrix for payloads of size 4096 bytes (4 packets), and 16384 bytes (16 packets). We can see clear improvements in the results by comparing the entries of the confusion matrix obtained from payloads of size 1024 bytes in table I, with the confusion matrix obtained from payloads of size 4096 bytes in table II, and 16384 bytes in table III. For example, note that the ZIP format is predicted correctly with a rate of 53%, and 73.83%, with payloads of 4096 and 16384 bytes respectively, as opposed to 33% with just 1024 bytes.

V. Discussion

As evident from the results above, the accuracy of the method increases with the size of payload considered during classification. Is it feasible to capture 4096 bytes of a flow, from a sample of network traffic? More clearly, suppose we collect 500 packets from a network is it likely to have at least 4096 bytes for each flow in the sample? In order to evaluate the viability of such an approach we captured around 100 million TCP packets over a 3 hour period in our campus edge router. Discarding the ACK packets that carry no payload the average size of a packet is 1188 bytes. The histogram of packet sizes could be seen in log scale in figure 5.

We also need to know how many packets need to be captured, to get the required number of bytes in sequence. To that extent, we take 100 packets from the collected traffic, exclude packets with size of less than 70 bytes, and computed the number of packets that belong to the same flow. Thus we obtain a histogram for the number of packets which are from the same flow. This process is repeated 100 times and the resulting histograms are averaged. The same procedure is repeated for 500, and 1000 sampled packets. The averaged histograms are shown in figure 6.

For example in order to get payload of size 4096 bytes, one needs to capture $\frac{4096}{1188} \simeq 4$ packets. Furthermore, a recent study of network traffic at a Tier-1 ISP shows only a

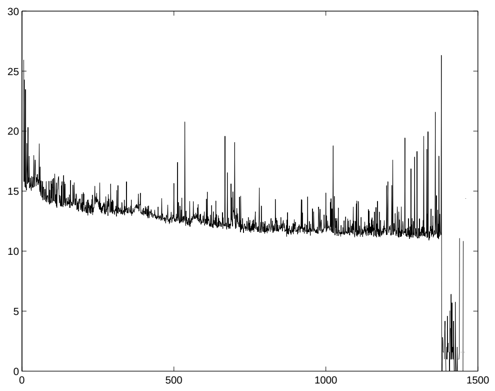

Fig. 5. Histogram of payload size for TCP packets. The y-axis is in log scale.

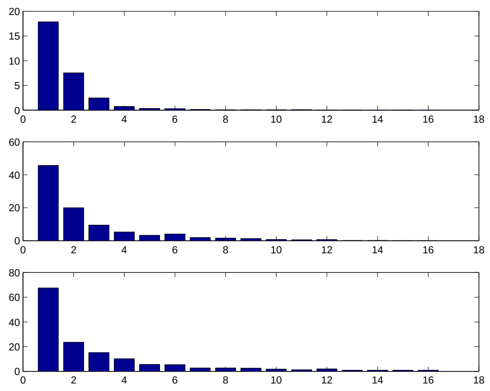

Fig. 6. Histogram of number of packets from the same flow. Top plot is for 100 packets being captured at once. Middle and bottom plots are for 500, and 1000 packets being captured at once.

fraction of (3.14%) TCP traffic is ever out of sequence [19]. Therefore, from the histograms in figure 6, we infer that capturing 500 packets with high probability yields at least 4 packets from the same flow and in sequence. Therefore, this method can is a viable alternative to that of monitoring every single packet in the network for flow content attribution using header information.

VI. Conclusion and Future Work

In this paper we proposed a novel problem, Flow Content Characterization, which attributes content types for network traffic on a per flow basis. We presented a method that can attribute flow content types for network flows based on payload and presented simulation results.

Improving the accuracy of the overall system for lower values of payload is part of our future work. One approach we are looking into is to exclude irrelevant or maybe contradictory features by using a feature selection algorithm. Furthermore, we are also interested in implementing a prototype system to be used on large networks.

TABLE I

Confusion matrix of flow content characterization using payload of size 1024 bytes (or equivalent to 1 packet)

	Predicted							
	Txt	Bmp	Wav	Zip	Jpg	Mp3	Mpg	Enc
Txt	**97.5**	1.83	0.33	0.17	0	0	0.17	0
Bmp	6.17	**88.17**	4.50	0.17	0.17	0.17	0.67	0
Wav	0	7.67	**79.17**	1.83	1.67	6.50	2.83	0.33
Zip	0.17	0.33	0.67	**33**	16.17	9.67	2.5	37.5
Jpg	0	0.50	0.83	30.33	**40.67**	8.67	5.67	13.33
Mp3	0	0	0.17	5.83	4.67	**82.33**	4.17	2.83
Mpg	0	1.67	1.5	2.5	6.83	9.5	**77**	1
Enc	0	0	0	15.83	3.33	4.67	0	**76.17**

TABLE II

Confusion matrix for flow content characterization using payload of size 4096 bytes (or equivalent to 4 packets).

	Predicted							
	Txt	Bmp	Wav	Zip	Jpg	Mp3	Mpg	Enc
Txt	**96.83**	2.5	0	0.67	0	0	0	0
Bmp	3.17	**91.67**	3.33	0.5	0.67	0.17	0.50	0
Wav	0.5	6.67	**82.17**	0.67	2	7.5	0.5	0
Zip	0.67	0.33	1.33	**53**	12	3.83	1.17	27.67
Jpg	0	0.33	0.67	12.50	**77.67**	4.33	3.33	1.17
Mp3	0.33	0.17	0.33	3.17	3.17	**90.50**	2.33	0
Mpg	0.67	2	1.5	1	4.67	6	**84.17**	0
Enc	0.17	0	0	12.67	0	0.33	0	**86.83**

TABLE III

Confusion matrix for flow content characterization using payload of size 16384 bytes (or equivalent to 16 packets).

	Predicted							
	Txt	Bmp	Wav	Zip	Jpg	Mp3	Mpg	Enc
Txt	**96.33**	2	0.67	0.83	0	0.17	0	0
Bmp	2.83	**91**	3.67	1.17	0.5	0.17	0.5	0.17
Wav	0.17	3.17	**88.33**	1.33	0.83	5.67	0.5	0
Zip	0.67	0	0.17	**73.83**	6	1.5	1.17	16.67
Jpg	0.17	1.33	0.83	4.17	**89.83**	2	1.67	0
Mp3	0	0.67	1.17	0.83	0.83	**95.83**	0.67	0
Mpg	0.83	2.33	0.83	0.67	2	2.67	**90.67**	0
Enc	0	0	0	2.33	0	0	0	**97.67**

References

[1] J. Reynolds and J. Postel, “Assigned numbers,” in *IETF RFC 1700*, October 1994.

[2] K. M. C. Tan and B. S. Collie, “Detection and classification of TCP/IP network services,” in *Thirteenth Annual Computer Security Applications Conference*, (San Diego, California, USA), pp. 99–107, December 1997.

[3] J. P. Early, C. E. Brodley, and C. Rosenberg, “Behavioral authentication of server flows,” in *Nineteenth Annual Computer Security Applications Conference*, (Las Vegas, Nevada, USA), pp. 46–55, December 2003.

[4] S. Kode, J. Maheswary, M. Nandwani, and S. Suresh, “Traffic characterization for heterogeneous applications,” in *Technical Report*, May 2001.

[5] S. Staniford-Chen and L. Heberlein, “Holding intruders accountable on the internet,” (Oakland), Proceedings of the 1995 IEEE Symposium on Security and Privacy, 1995.

[6] Y. Zhang and V. Paxson, “Detecting stepping stones,” in *Proceedings of the 9th USENIX Security Symposium*, (Denver, Colorado, USA), August 2000.

[7] H. Debar, M. Dacier, and A. Wepsi, “A revised taxonomy for intrusion-detection systems,” IBM Research Report, 1999.

[8] S. Kumar and E. H. Spafford, “An application of pattern matching in intrusion detection,” Purdue University Technical Report CSD-TR-94-013, 1994.

[9] K. Ilgun, R. A. Kemmerer, and P. A. Porras, “State transition analysis: A rule-based intrusion detection approach,” IEEE

Transactions on Software Engineering, March 1995.

[10] J. Frank, "Artificial intelligence and intrusion detection: Current and future directions," In Proceedings of the 17th National Computer Security Conference, 1994.

[11] H. S. Javitz and A. Valdes, "The sri ides statistical anomaly detector," In Proceedings of the IEEE Symposium on Research in Security and Privacy, 1991.

[12] P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling responses to anomalous live disturbances," In Proceedings of the National Information Systems Security Conference, 1997.

[13] V. Paxson, "Bro: A system for detecting network intruders in real-time," 7th Annual USENIX Security Symposium, January 1998.

[14] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," In Proceedings of the ACM Conference on Computer and Communication Security, November 1999.

[15] S. Axelsson, "Research in intrusion-detection systems: A survey," Technical Report No 98-17, December 1998.

[16] J. Mendel, "Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications," *IEEE Proceedings*, vol. 79, pp. 278–305, March 1991.

[17] V. Vapnik, "The nature of statistical learning theory," *Springer-Verlag, New York*, 1995.

[18] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[19] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a tier-1 ip backbone," in *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment*, (Marseille, France), pp. 113–114, ACM Press New York, NY, USA, November 2002.