Boundary Detection in Tokenizing Network Application Payload for Anomaly Detection

Rachna Vargiya and Philip Chan
Department of Computer Sciences Technical Report CS-2003-21
Florida Institute of Technology
Melbourne, FL 32901
rvargiya@fit.edu and pkc@cs.fit.edu

Abstract

Most of the current anomaly detection methods for network traffic rely on the packet header for studying network traffic behavior. We believe that significant information lies in the payload of the packet and hence it is important to model the payload as well. Since many protocols exist and new protocols are frequently introduced, parsing the payload based on the protocol specification is time-consuming. Instead of relying on the specification, we propose four different characteristics of streams of bytes, which can help us develop algorithms for parsing the payload into tokens. We feed the extracted tokens from the payload to anomaly detection algorithm. Our empirical results indicated that our parsing techniques can extract tokens that can improve the detection rate.

1. Introduction

Motivation: Traditional intrusion detection systems use *misuse/signature detection*, which models known attacks, and generally cannot detect novel attacks. *Anomaly detection* models normalcy and identifies deviations, which potentially can be novel attacks. During training, network anomaly detection models the normal patterns of network traffic. During detection, scores are assigned to anomalous events and significant anomalies cause alerts indicating possible attacks. Existing anomaly detection techniques usually reply on information derived only from the packet headers; however, this is not sufficient since more sophisticated attacks involve the application payload. Parsing packet headers is relatively simple as there are few commonly used protocols such as IP, TCP, UDP, and ICMP. However, for application payloads, parsing is more challenging due to the large number of application protocols available and relatively frequent introduction of new protocols. *Hard coding* the parser for each application protocol could be time consuming, particularly when the protocols are complicated. Furthermore, updates to existing protocols or introduction of new protocols will require additional efforts.

Problem statement: We desire to parse application payload into tokens without explicit knowledge of the application protocols. Given a set of exemplar payloads, an algorithm learns a model that can parse the payloads into “meaningful” tokens. Furthermore, the algorithm needs to be independent of the protocols and a model can be built for each protocol using the same algorithm. The extracted tokens can then be used as attributes for modeling normal traffic for anomaly detection (the same techniques can also be used to identify tokens for misuse detection as well, but anomaly detection is the focus of this paper).

Approach: We propose four characteristics of relevant tokens in a continuous stream of bytes, and based on them, design algorithms that propose possible boundaries for tokens. We use these characteristics individually and in combination to estimate boundaries. The sequence of bytes between the two successive boundaries is considered a token which can be used to model the behavior of the payload. The characteristics are based on Boundary Entropy, Frequency, Augmented Expected Mutual Information, and Minimum Description Length. These characteristics do not depend on any particular property of a protocol.

Contributions:

- We describe four algorithms based on the characteristics mentioned above, and apply them to parse the payload to extract tokens from network traffic.
- We also explore techniques using more than one such characteristic in combination.
- We discuss four evaluation techniques to evaluate such tokens.
- We demonstrate that the tokens found by our algorithm can improve the detection rate of the LERAD anomaly detection algorithm.

Organization: The next section, Section 2, discusses the related work. Section 3 details the four characteristics and the associated algorithms. In section 4 we discuss results obtained from our experimental evaluation, and finally we conclude in Section 5.

2. Related work

Varieties of approaches have been adopted for the boundary detection problem. Some of them are unique and achieve interesting results.

One of the early, works include that described in Forrest et al. [1] They used fixed length patterns to represent the process model and used it for intrusion detection purpose. However, a main limitation of this approach is that there is no rationale for selecting the optimal pattern length, which has a major influence on the detection capabilities of the intrusion detection system. In addition, it uses fixed length patterns, which makes it a difficult task to select the optimal pattern length. Long patterns are expected to be more process specific than short patterns. Our approach is independent of such a parameter like length and hence overcomes this problem.

Wespi et al. [9] use Teiresias algorithm in combination with a pattern reduction algorithm to construct patterns. All maximal variable length patterns contained in the set of training sequences are determined and a reduction algorithm is applied to prune the entries in the pattern table. Their pattern-matching algorithm returns the g groups of consecutive uncovered events and the length (l (i)) of each of these groups. The greater the length l (i), the more likely it is that an intrusion is observed.

Liao et al. [2] use k-Nearest Neighbor classifier to characterize program behavior as normal or intrusive depending on the short sequences of system calls. Even though the computation required for this technique is reduced, it is unable to detect attacks that consist of abuse of a legal attack, e.g. Process table attack. Some text categorization work is also done by Dumais et. Al [10].

Jiang et al. [6] consider both Intra pattern and Inter pattern anomalies. They provide a pattern extraction algorithm to identify maximal patterns. Then they use a Pattern overlap relationship module where adjacency lists are formed from patterns in which overlap relationship between patterns is stored. Pattern adjacency lists are then traversed at real time to identify both intra pattern and inter pattern mismatches. Significant deviations from the normal behavior cause the module to raise alerts.

Valdes [5] proposes a system that maintains a library of patterns that may be initially empty. When a pattern is observed, its similarity with respect to other patterns in the library is observed. If it matches one or more stored patterns above a configurable threshold, then the new pattern is considered to belong to the class of the best matching. However, Their approach works only with a small set of alphabet and small number of actual observed patterns.

Michael [8] uses suffix trees of a fixed height to find frequent occurring sequences of system calls. Very frequent sequences are replaced with meta-symbols, resulting in a more compact representation of the system calls. Based on the revised vocabulary, a regular language is learned to represent the normal behavior of system call traces.

One of the algorithms, SEQUITUR proposed by Manning and Witten [4] gives a technique for parsing the text, which is our first step. It is based on the principle that phrases, which appear more than once, can be replaced by a grammatical rule that generates that phrase. The rule generated is different from conventional grammar since the rules are not generalized and they generate only one token.

Another algorithm proposed by Cohen et al. [3], called VOTING EXPERTS consists of experts that evaluate the features of the episodes, namely Boundary Entropy and Frequency, and votes for boundaries in the corpus based on these features. A window is passed through the corpus and each location garners 0 or 1 vote from each expert. The location with least boundary entropy and highest frequency receive votes from the two experts respectively. The drawback of their technique is that their votes are binary; the confidence in a particular boundary cannot be indicated.

3. Approach

Our approach consists of parsing the payload and extracting tokens providing some information about the payload, and using these tokens to model the network behavior for anomaly detection. This approach demands is a good algorithm to parse the payload. There are characteristics that can categorize bytes belonging to some relevant token. Hence, these characteristics can be exploited to detect boundaries in a continuous stream of characters. By extracting the token between two boundaries, we can derive the set of bytes belonging to the same token.

Our approach is inspired by VOTING EXPERTS [3] as in features are used to detect boundaries. However, there are many differences in the details of the approaches. Not only have we created more experts for casting votes and combined those experts, we also intend to make a system that allows certain feature to cast multiple votes, depending on how strongly that feature believes that a boundary exists at that location. In VOTING EXPERTS, each feature was independent and could cast only binary votes, whether the feature strongly suggested a boundary or there was just a slight indication of the same. Since the features we use to assess potential boundaries are statistical, our approach is independent of the language or in our case, independent of the protocol of the application layer. Hence, our technique is domain independent.

The general working of the algorithm includes a window of arbitrary size (given as an input), say *w*, which is slid through the corpus to be segmented. At each instant, *w* bytes from the corpus are observed. Each feature evaluates the value for each possible boundary within the window, and decides whether the value is good enough for a boundary or not. If yes, a vote is cast, otherwise the window simply slides one character forward, examining again the token of length *w*, differing in one byte from the previous token. Two parses are required for this approach on the corpus, first to evaluate the feature value for each possible token, second to compare the various possible boundary locations based on the evaluated feature value and to assign votes.

There are four features used to cast votes in our model. Two of them are similar to the ones in VOTING EXPERTS: Boundary Entropy and Frequency. The other two are Augmented Expected Mutual Information (AEMI) and Minimum Description length. Finally, we propose techniques by combining some or all of them. Each one is discussed in some detail below.

3.1 Boundary Entropy

The entropy in patterns exhibits a trend that is exploited in this technique. It starts with a relatively high value, then drops as we go further, and peaks at the end of a valid word. This is because entropy gives us the uncertainty or degree of randomness in a system. When we see the first few characters of a word, it is difficult to predict what the word is. E.g., given the character ‘W’, it is difficult to say what the word is. It could be ‘what’, ‘where’ or any other such word. Hence, the entropy after ‘W’ is relatively high. However, as we move further, the uncertainty drops. E.g., if we have the token ‘Wha’ then we know that the word probably is ‘What’. At the end of the meaningful token, the entropy peaks. This is because it becomes very uncertain what the following word is going to be, and hence what the next character should be. Like in our example, any word could follow ‘What’, so it is difficult to say what the next character will be.

We exploit this property to create an expert to detect boundaries. The entropy at each possible location is calculated using the formula, similar to voting experts, i.e.

$$-\sum P(x)\log P(x) \tag{1}$$

$P(x)$ is the probability of the byte x following the current window. (More precisely, $P(x)$ is actually $P(x|s)$, where s is the sequence of bytes in the window of size w.) The entire expression gives us the uncertainty of the byte following the token in the window. During the first parse, the window is moved across the file and the byte following the window is noted and $P(x)$ is estimated. In the second parse, the window is moved again and Boundary entropy at the end of each window is calculated using the formula given above.

The positions that have the maximum entropy get a vote from the expert. However, since we desire to signal only boundaries with reasonable confidence, we introduce a threshold that suppresses votes from boundaries with low entropy values. We use the average boundary entropy of the corpus to be the threshold. To allow fair voting among experts, the boundary entropies are normalized before votes are cast. The votes cast are proportional to the number of standard deviations away from the mean value.

3.2 Frequency

This method computes the frequency of each token that occurs in the corpus. The most frequent set of tokens are assumed to be valid tokens and boundaries are assigned at the ends of such tokens. As the window moves forward, the frequency of each possible token of length 1 to the window size, within the window, is calculated. E.g. if the window consists of “examsare”, then the frequency of ‘e’, ‘ex’, ‘exa’ and so on is calculated. Boundary is voted at the end of the most frequent token. The votes given are proportional to the number of standard deviations that the frequency of the token is away from the mean.

The window is moved through the corpus and each token formed is counted. Hence, at the end of the parse, we have a list of all possible words, which the window may consist of, and their frequency. Generally, in most domains, there is a relationship between the length and frequency of patterns. Short patterns tend to be more common than the long ones. E.g., ‘t’ would be more common than ‘the’ even though ‘the’ is a valid word and ‘t’ is not. We want to compare how unusual a pattern is, not just how frequent it is. Therefore, comparing the frequencies of short patterns with that of long patterns would not be appropriate. To accommodate this, we normalize the frequencies of the tokens. We subtract the sample mean from the value and divide by the sample standard deviation.

3.3 Augmented Expected Mutual Information (AEMI)

A lot of information can be gathered about a character based on the context it appears in. Generally, the concept of mutual information is used to evaluate the relationship between two events. Mutual Information can estimate the likelihood of the occurrence a token given some other token. E.g. talking about food, given that we have seen ‘POP’ it is very likely that the next word would be ‘CORN’. Hence, this approach is based on co-occurrence of tokens: if two tokens appear together frequently, they are probably part of the same word. Mutual information is given by:

$$MI(a,b) = \log[P(a,b)/(P(a)P(b))] \qquad (2)$$

In other words, MI gives us the reduction of uncertainty in presence of ‘b’ in the window if presence of ‘a’ is known (or vice versa). However, this metric only considers the presence of both the words but not the absence of either of them. That is, it does not consider what the probability of seeing one token in the absence of the other. This leads to misinterpretations if the token whose occurrence is being measured is highly frequent. E.g., we would expect that ‘pop-corn’ is more correlated than ‘is in’, however since ‘is’ is relatively more common. This would lead to a high MI value. The presence of one token without the other one counts for adverse correlation and proves to be counter evidence. To capture this idea, Augmented Expected Mutual Information (AEMI) is used. AEMI appropriately incorporates the counter evidence. It is given by:

$$AEMI(A,B) = P(a,b)MI(a,b) - P(\neg a,b)MI(\neg a,b) - P(a,\neg b)MI(a,\neg b) \qquad (3)$$

Equation 3 sums the supporting evidence and subtracts the counter evidence. *A* is defined as the event of the first token, and *B* is the event of the token following the first one. Higher values of AEMI indicate that *A* and *B* are probably part of the same word. We only consider three cases with each pair of tokens. occurrences when both the tokens appear together, when token *a* appears without token *b*, and token *b* appears without token *a*. The case when neither *a* nor *b* appears is disregarded since it does not really present much information about whether *a* and *b* co-occur or not. The window is again moved across for each possible boundary, the left and right sub tokens are considered. E.g. If a window contains “abcdef” we consider left and right sub tokens ‘a’ and ‘bcdef’, then ‘ab’ and ‘cdef’, then ‘abc’ and ‘def’ and so on. Then for each set of left and right sub tokens within a window, AEMI value is computed and compared. For each window, the location with the minimum AEMI value suggests a boundary, and the expert gives votes proportional to the standard deviations from the average AEMI.

3.4 Minimum Description Length

In coding theory, tokens that are more frequent are assigned a shorter code so that the overall coding length is minimized for a message with multiple tokens. Minimum Description Length (MDL) assumes a perfect encoding and measures the fewest number of bits necessary to encode a message. We calculate the description length per byte of a token by:

$$MDL = \sum_{i \in \{left, right\}} -\lg P(t_i / | t_i |) \qquad (4)$$

where t_i denotes the two tokens on the left and right of the possible boundaries, P(t_i) is the probability of t_i and | t_i | is the length of t_i in bytes. The assumption is that if we were to compress the file, we would assign minimum number of bits to the most frequently occurring token, hence, it would have the minimum length. –lg [P(t_i)] gives us the number of bits used for t_i, dividing it by the length of t_i, | t_i |, gives us the number of bits per byte of the token or its description length.

The preprocessing is similar to that in AEMI. The first parse is used to look at all the left and right sub tokens and compute the probability of seeing those two tokens. This probability is then used in the second phase, which computes the sum of the description lengths of left and right sub tokens. As the window slides over the data, the boundary that yields the shortest coding length is voted as the boundary and the number of votes is again proportional to the number of standard deviations from the average coding length.

3.5 Combined Approach

All the approaches discussed so far use single expert to suggest boundaries. We desire to design a model that combines the opinions from all the experts and then decides upon the boundary. In this method, we allow each expert to run a scan on the file and decide where to vote, and how much to vote. Votes from each expert are normalized, as some approaches may tend to assign more votes than the others do. These votes are then combined to locate positions most strongly suggested as the boundary after consideration by all the experts. A list of votes from all the experts is gathered. This list is normalized so that the votes from each expert indicate the confidence of the expert and are on the same scale. In order to normalize the list, the standard deviation of the votes is computed and each value in the list is divided by the standard deviation. This scales the value with respect to the other values in the list. In order to give more weight to a particular expert, the votes from this expert can be increased by a certain factor. For each boundary, the final votes from each expert, after normalization, are summed. A threshold is set depending on the final set of votes. A boundary is placed at a certain position if and only if the votes at that position exceed the threshold.

We tried combining all the algorithms and then combining the strongest algorithms, frequency and minimum description length.

3.6 Anomaly Detection

Once we have placed the boundaries according to our experts, we can easily extract the meaningful tokens from the file. Our anomaly detection system, LERAD [7], forms rules based on attributes picked from the network data (including header and payload). Currently, it uses tokens from the payload that are "space" separated. Instead, we modify it to use tokens that are separated by boundaries identified by the algorithms discussed above.

4. Experimental Evaluation

4.1 Evaluation Criteria

We present four different types of evaluations depending on various attributes that would indicate the "meaningfulness" of the tokens retrieved in the output file.

The first evaluation, Evaluation A, is based on how many words, present in the input file, were we able to retrieve in the output file with the boundaries placed at positions suggested by the expert. All space or punctuation separated tokens are assumed to be "meaningful" tokens. This evaluation works for text-based protocols only. It doesn't work for non-text based protocols because bytes that represent "spaces" usually do not exist. Moreover, Evaluation A only approximates how "tokens" are defined. E.g. a file name could consist of '/' to denote the path of the file. The entire path should be considered as one token, however since our evaluation would consider '/' as space, it will consider each directory a unique token. Also, for hyphenated words, even though they would logically be the same token, this evaluation would evaluate them as being separate tokens. Based on the space-separated words, we report the percentage of words recovered by our methods.

The second evaluation, Evaluation B, is similar to the first evaluation except that it looks for certain keywords that are characteristic of the particular application protocol. These keywords are collected from the specification of each protocol (Request for Comments or RFC). However, this evaluation is limited to text-based protocols for the reasons mentioned above and is an approximation since tokens between two keywords are not specified. Based on the known keywords, we report the percentage of keywords recovered by our methods.

The third evaluation, Evaluation C, calculates the entropies of the output files. The motivation for this evaluation is that if the expert was successful and it found most of meaningful tokens, then the tokens should be repeated often in the output file, leading to less randomness in the output file and therefore to lower entropy values for the file. Thus, in our evaluation, the lower the entropy value of the output file, the better is the feature or expert. This evaluation is independent of any text-based assumptions and hence can be used for all kinds of ports. It gives a good estimate of the output file. It can be used to compare the performance of an approach on any kind of protocol.

The fourth evaluation, Evaluation D, is the detection rate evaluation, which is the most important evaluation, while Evaluations A-C are intermediate approximations. We measure the number of detections at various false alarm rates and compare the performance of the original LERAD with LERAD using tokens extracted by our proposed methods.

4.2 Evaluation Data and Procedures

The proposed methods were evaluated using the 1999 DARPA Intrusion Detection Evaluation Data Set [11]. The test bed involved a simulation of an air force base that has machines that are under frequent attack. These machines comprise of Linux, SunOS, Sun Solaris and Windows NT. Various intrusion detection systems have been evaluated using this test bed. It comprises of three weeks of training data obtained from network sniffers, audit logs, nightly file system dumps and BSM logs from Solaris machine that trace system calls and two weeks of testing data. Weeks 1 and 3 of the data are attack free while various attacks are present in Weeks 4 and 5 of the data.

For our first three evaluations, where we compute the number of words retrieved, number of keywords retrieved, and the entropy of the output file, we use only the data from Week 3. The reason for using week 3 for evaluations A, B, and C is that Weeks 4 and 5 are for testing only and we do not want to have the advance knowledge of which tokenization methods work better in Weeks 4 and 5. We used the first four days of Week 3 for training and the last three days for testing. This gives us an estimate of how predictive the approaches are, and how well they would perform on unseen data in the network traffic. We studied the ports with the most traffic and results from these ports are reported. The window size was a parameter set to six, which was experimentally observed to be the best value.

The anomaly detection system LERAD [7] works in three phases. In the first phase, it samples training pairs to suggest rules. In the second and third phases, it removes redundant rules and rules that generate alarms on attack free traffic respectively. LERAD learns rules based on 23 attributes taken from the TCP header and the payload. First 15 attributes are picked from the packet header and the remaining eight are picked from the payload. LERAD, originally picks the first eight space separated tokens from the payload--space as boundary is not applicable to non-text protocols. We replace these eight space separated tokens with the more intelligently found boundary separated words from our approaches. For Evaluation D, we use Week 3 for training and Weeks 4 and 5 for testing.

4.3 Experimental Results and analysis

We present results for six different approaches, four approaches being the results of the four algorithms independently, fifth being the combination of all the algorithms and fourth being the combination of two of the strongest algorithms, Frequency and MDL. The reason for combining Frequency and MDL is that, from our experience with this data set, they provide maximum coverage and complement each other.

4.3.1 Evaluation A: Space Separated Tokens

Table 4.3.1 Evaluation A: % of Space-Separated Tokens Recovered

Method	Port #25	Port #80	Port#21	Port #79
Frequency	31	28	13	99
Min Desc. Length	7	6	3	25
AEMI	9	5	4	32
Boundary Entropy	3	2	1	9
All 4 experts	12	13	5	12
Freq + MDL	30	36	21	81

Table 4.3.1 reports the results of all the approaches on popular ports with text-based protocols, SMTP (25), HTTP (80), FTP (21) and Finger (79), based on Evaluation A. For all these ports, Boundary Entropy gives the poorest results. Frequency performs the best for SMTP and Finger, however Freq + MDL performs best for HTTP and FTP. On qualitative analysis, Freq + MDL seems to give a more consistent output with long relevant tokens. Hence, we suggest that Freq + MDL together gives the best results followed by the single approach of Frequency. The model of all the algorithms combined follows these two techniques. MDL performs better than Frequency when trained and tested on the same set, however it is not very predictive. Frequency on the other hand, is highly predictive. Hence, these two algorithms tend to find different kinds of words. When combined they give maximum coverage and hence best results.

4.3.2 Evaluation B: Keywords in RFCs

Table 4.3.2 Evaluation B: % of Keywords in RFCs Recovered

Method	Port#25	Port#80	Port#21
Frequency	31	28	40
Min Desc. Length	7	6	1
AEMI	9	5	2
Boundary Entropy	3	2	2
All 4 experts	12	13	21
Freq + MDL	40	36	59

Table 4.3.2 reports the results for all the methods based on Evaluation B. Results for port #79 are absent since no keywords were available for port 79. Here again Frequency + MDL performs the best for ports #80 and #21. However, for port #25, frequency alone performs better. The ranking of the algorithms remains the same and reinforces our conclusions from the previous evaluation.

4.3.3 Evaluation C: Entropy

Table 4.3.3 Evaluation C: Entropy of Output

Method	Port#25	Port#80	Port#21	Port#79	Port#1023	Port#22
Frequency	9.19	5.11	5.17	3.78	0.86	5.79
Min Desc. Length	8.61	5.26	5.50	1.43	0.77	8.61
AEMI	8.66	5.74	9.23	6.27	1.10	7.32
Boundary Entropy	7.89	5.36	6.79	2.63	0.96	7.75
All 4 experts	9.52	5.07	5.36	6.32	1.39	5.74
Freq + MDL	7.94	4.98	9.04	4.31	1.91	8.32

Table 4.3.3 reports the results of all the approaches based on Evaluation C on four text based and two non text based ports, Smtp (25), Http (80), Ftp (21), Finger (79), SSH (22), and TCP Reserved (1023). This evaluation compares the schemes on both text based as well as non-text based ports and allows us to compare the techniques without any bias. The relative values vary for different ports. For port #25 Boundary Entropy gives the best results, however for ports #79 and #1023, Minimum Description Length gives lowest entropy. Frequency gives lowest entropy values for port #22, Freq + MDL and the combination of all four methods achieve the lowest entropy for #80 and #21 respectively. Since

all techniques are very close in this evaluation, it is difficult to say which technique is best for all ports based on this evaluation only. However we can make port specific conclusions like for port #80, Freq + MDL is the best technique.

4.3.4 Evaluations on Combined models

Table 4.3.4 Results from Additional Ports for Freq + MDL and ALL

Port #	Evaluation A % Words Found		Evaluation B % Keywords Found		Evaluation C Entropy	
	Frq+MDL	ALL	Frq+MDL	ALL	Frq+MDL	ALL
23	13	7	5	3	7.88	8.08
113	43	20	--	--	4.45	5.18
515	38	14	--	--	7.66	7.27

Since Frequency + MDL and the model combined of all algorithms have the potential of giving better boundaries indicated by evaluations A-B and evaluation C respectively, we performed experiments on the remaining ports with these two techniques. *Table 4.3.4* reports results of the two models, Frequency + MDL and the combination of all algorithms on additional ports, for all three evaluations. Based on these results, it is evident that Frq+MDL performs better than the model combining all four approaches. Even though Frq+MDL performs very well, the inclusion of the other two techniques weakens the model. This could be attributed to the probability that with the inclusion of AEMI and BE the model gets confused and results deteriorate. Boundary Entropy in particular attempts to vote at too many positions and lowers the performance.

4.3.5 Evaluation D: Detection Rate

Based on our first three evaluations, we picked the most promising technique for our fourth and most important evaluation. From the previous evaluations, it was obvious that certain techniques may be better depending on the port. However, the model consisting of Frequency and Minimum Description Length gave a good performance consistently. Thus, we decided to perform our final evaluation on this technique.

Table 4.3.5 Detection Rate for Space Separated LERAD and Boundary Separated LERAD using Freq + MDL tokenization

PORT#	10 FP/day		100 FP/day	
	Space-Separated	Boundary-Separated	Space-Separated	Boundary-Separated
20	2	2	4	5
21	14	16	14	17
22	3	3	3	3
23	13	14	13	14
25	15	16	16	16
79	3	3	3	3
80	10	10	11	13
113	2	2	2	2

LERAD forms conditional rules that are used to test tuples from test data. The alarms generated were evaluated for two different allowed false alarm rates – 10 and 100 per day respectively. The results, reported in *Table 4.3.5,* indicate some improvements in the total number of detections for both text based and non-text based protocols. Port #20 shows an improvement of one detection when the false alarm rate is set to 100 per day. Considerable improvement for port #s 21, 23 is observed for both false alarm rates. Port #25 and #80 also show an improvement of one attack each at false alarm rates of 10 per day and 100 per day respectively. For other ports where the results are comparable, we suggest two possible reasons. Firstly, the training data for these ports was not sufficient for the experts to cast vote during the testing phase. In addition, for certain ports, it never generated any rules based on the tokens from the payload—LERAD did not find the payload tokens to be indicative of normal behavior. In such cases, even if tokens that are more meaningful were extracted by our algorithms would not affect the results.

We also performed experiments using a combined model of all the ports instead of using port specific data. Even then LERAD with space-separated tokens finds 36 attacks in week 4 data as compared to 38 attacks detected if boundary separated tokens are considered. The false alarm rate was 10 per day for this result. On increasing this rate to 100 per day, the former still detects 36 attacks while the latter detects 39. Data for week 5 was not used for these results and experiments are still being conducted to evaluate this technique on week 5 data of the DARPA data set.

5. Concluding Remarks

In this paper, we present the four algorithms based on characteristics mentioned above, and apply them to parse the payload to extract more information about the traffic. The results of each of those techniques applied independently and then applied in various combinations based on these evaluations are given. According to the experimental results obtained from the DARPA 99 dataset, we observed that Frequency and MDL are two strong experts individually and achieve good results. MDL works even better when training and testing sets are more similar. Frequency is highly predictive and does well on different training and testing sets. When combined, the model formed by combining Frequency and MDL is found to be the strongest. Combining all four methods does not do as well as Frequency + MDL. This payload parsing method, when applied to the LERAD anomaly detection algorithm, leads to an increase in the detection rate in two configurations: individual LERAD model per port or single LERAD model for all ports.

Our goal is to use these approaches to improve the features used by the anomaly detection algorithm LERAD [7]. Another improvement can be made by instead of using the first eight boundary separated tokens, the tokens which are likely to give maximum information should be used. This property of the tokens can be measured by again looking at features like frequency, AEMI and so on. Of the words that are retrieved in the output, the ones with maximum feature value are likely to give us maximum information. In addition, the first two evaluations, A and B are not very accurate. They are more suited for algorithms that intend to parse natural language. We will try to build more evaluations that can give us a better idea of our output before feeding it into the IDS. Entropy is one such evaluation and we should try to build more. Especially for non-text based protocols, this is important. We will also try to integrate our technique, i.e. incorporating information from the payload to more intrusion detection systems.

Acknowledgement

This research was partially supported by DARPA (F30602-00-1-0603).

6. References

[1] Steven A. Hofmeyr, Stephanie Forrest and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security,* 1998.

[2] Yihua Liao, V. Rao Vemuri. Using text categorization techniques for intrusion detection, In *Proc. 11th USENIX Security Symposium,* 2002.

[3] Paul Cohen, Brent Heeringa, and Niall Adams. An unsupervised algorithm for segmenting categorical time series into episodes, *IEEE International Conference on Data Mining,* 2002.

[4] Nevill Manning, C.G., Witten, I.H. Identifying hierarchical structure in sequences: A linear time algorithm, *Journal of Artificial Intelligence Research*, 7, 67-82.

[5] Alfonso Valdes, Detecting Novel Scans Through Pattern Anomaly Detection, In *Proc. DISCEX, 2003*

[6] Jiang N., Hua K., and Sheu S. Considering Both Intra-pattern and Inter-pattern Anomalies in Intrusion Detection, In *Proc. of ICDM,* 2002.

[7] Matthew V. Mahoney, Philip K. Chan, Learning Models of Network Traffic for Detecting Novel Attacks, *Technical Report CS-2002-08, Florida Institute of Technology.*

[8] C. C. Michael, Finding the Vocabulary of Program Behavior Data for Anomaly Detection, In *DISCEX,* 2003.

[9] Andreas Wespi, Marc Dacier, and Herve Debar. Intrusion detection using variable length audit trail patterns, In *Proc. of RAID,* 2000.

[10] Susan Dumais, John Platt, David Heckerman, Inductive Learning Algorithms and Representations for Text Categorization, In *Proc. of ACM-CIKM98,* 1998.

[11] R. Lippmann, J. Haines, D. Fried, J. Korba & K. Das. The 1999 DARPA Off-Line Intrusion Detection Evaluation, *Computer Networks*, 34(4), p579-595, 2000.