

# Payload-Based Anomaly Network Intrusion Detection

Wilmar Sulaiman

## **Abstract**

Payload is an essential feature to identify anomalous packet. PAYL[38], a 1-gram payload based network intrusion detection has been proven to be effective in detecting malicious attacks. In this thesis, we investigate the effect of having higher order  $N$ -gram under various resolutions. We evaluated our method by performing experiments over network records from 1999 DARPA IDS dataset [33]. Our evaluation shows that the accuracy of using 2-gram has improved, but lower resolution is not necessary better.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Literature Review</b>	<b>3</b>
2.1	Type of Attacks . . . . .	3
2.2	Type of Anomaly . . . . .	3
2.3	Network Anomaly Detection . . . . .	5
<b>3</b>	<b>Methodology</b>	<b>9</b>
3.1	N-gram payload byte frequency based length and port . . . . .	9
3.2	Mahalanobis Distance . . . . .	14
3.3	Incremental Learning . . . . .	15
3.4	Clustering . . . . .	15
3.5	Overview . . . . .	18
<b>4</b>	<b>Evaluation</b>	<b>19</b>
4.1	Evaluation of Darpa Dataset . . . . .	19
4.2	Experimental Setup . . . . .	21
4.3	Result . . . . .	24
<b>5</b>	<b>Discussion</b>	<b>32</b>
5.1	Result Discussion . . . . .	32
5.2	Performance Discussion . . . . .	33
<b>6</b>	<b>Future Work</b>	<b>35</b>
<b>7</b>	<b>Conclusion</b>	<b>37</b>

# 1 Introduction

It becomes evident that computer security is becoming more important in today's world. The Computer Emergency Response Team or CERT [2] identified there were 137,529 incidents and 3,780 new vulnerabilities in 2003. Although the number of new vulnerabilities starts to decline, the number of incidents has approximately doubled each year since 1997. The figure is consistent to our believe that most of vunerabilities are due to software error and the number of vulnerabilities starts to decline is due to most of software developer nowadays tend to use the latest programming language that has extra features to minimize the security threat. For example, buffer overflow vulnerabilities that's heavily exploited in C programming language are not found in later programming language. However due to significant growth of the computer network, the number of incidents has not declined. Attacks like worms and viruses can spread at an exponential rate through computer networks. For example, SQL Slammer worms in 2003 infected 75,000 machines in just 10 minutes [2]. Therefore the development robust and reliable Network Intrusion Detection is increasingly important.

Traditional signature-based network intrusion detection like SNORT[36] and Bro[28] have been widely used to detect known attacks. The idea is that once the attack methods have been discovered, the associate network traffic patterns are recorded and coded by security experts. This type of network intrusion detection has a major drawback;the inability to detect new types of attack because there is always a time gap between the exploit and the updated intrusion detection, thus leaving the time gap periods unsecured. Furthermore, there is a space limit that the signature-database can store. After all, since one particular attack could appear in many forms (polymorphism) thus leaving this strategy is not feasible for expansion.

A promising approach to detect novel attacks is to use anomaly detection approach. The idea originally came from Forrest et al [34]. Forrest argued that our own immune system can provide far better ideas to improve network intrusion detection techniques. He made an analogy that our immune system often attacks transplanted organ from donor because the object is different in nature. As a result to control and reduce rejection, doctor often injected drugs. Forrest believed that the same observation can be made on the Intrusion Detection System, by looking at the normal traffic packet we could claimed that packet that deviates from the normal model can be classified as anomalous traffics. This method often incorporates various data mining techniques. However in most cases due to the nature of anomaly detection classification is somehow abstract,often it suffers from high false positive rate.

In this paper we will be following the same analogy i.e. we classify every-

thing that is different from the normal model as attacks. To achieve our goal we model the system by using 2 phases: *learning phase* and *detection phase*. During the learning phase we analyze the payload content by producing their byte frequency of distribution for each length packet. It is important to note that our detection phase is site specific, thus each site has different normal distribution. Furthermore our learning phase is also service specific because each services produces different payload's characteristic so computing them as single distribution might mislead the result. After computing the byte frequency from normal models, the anomaly detection phase begins. At this stage we compute the model by using Mahalanobis Distance. If the new model deviates from the normal model by some threshold we classify them as an attack. The idea is based on the assumption that attacks often appear as anomalous payload.

However unlike PAYL in [38] that used 1-gram distribution, in this thesis we investigated the effect of using higher order N-gram. We realized that using higher N-gram will result in higher complexity in both computational and space, therefore we used various resolution techniques and investigated the effect on the detection accuracy.

In this paper we present our payload based algorithm to detect malicious packets and not to prevent. Our goal here is to detect novel attack so we did not include any signature or pattern matching signature in our intrusion detection. The thesis is structure as follows. In section 2, we describe general overview of anomaly detection. We introduced some new concept in anomaly detection against 1999 DARPA dataset [32, 33]. In section 3, we described our network anomaly detection algorithm in detail. In section 4, we described the performance against the 1999 DARPA dataset. We also mentioned the coverage of our algorithm. In section 5, we discuss the result and what we thought about the intrusion detection. In section 6, we suggest some possible future directions of the investigation. We conclude the thesis in section 7.

## 2 Literature Review

In this chapter, we explain the required background knowledge and present some recent work on intrusion detection in particular anomaly detection. This includes an introduction to type of attacks and anomalous behaviors. This leads to a survey of various approaches anomaly detection schemes. Furthermore, we also explain various advantage and disadvantage of a particular technique.

### 2.1 Type of Attacks

There are many known exploits. Kendall [14] clustered them into four major categories :

- Denial of Service - an attack which prevents legitimate user to access the victim. This can be achieved by either make the system's resource busy or crash the vulnerable software.
- Probes - testing and gathering some information about the target machine.
- Remote to Local (R2L) - an attack in which an attacker that do not have login access can execute local command in the victim machine.
- User to Root (U2R) - an attack in which attacker with login access can gain administrator access by exploiting services.

The first three attacks often exploit network implementations [19]. However the latter category which is User to Root (U2R) attacks often exploit incorporated local vulnerabilities. Clearly in this case we are more interested in the first three types of attacks.

### 2.2 Type of Anomaly

In order to building the right anomaly detection techniques, we believed that it is important to understand the nature of anomalous traffics. Matt Mahoney [25, 22, 23, 21] identified 5 types of anomalous packets:

- User Behavior Anomaly  
Hostile packets might come from anomalous activity from the user. For example most of U2R (User to Root) attacks involved 2 steps. The first step is log on to the system, and then the users upload the exploit from

the FTP (File Transfer Protocol). The step can be seen as an anomaly because most of the FTP connection involved sending data to the user (download - in user perspective) not upload. It suggests that the user play an important role of determining the step of the attack. It is possible to conduct this operation using different method for example it might be the case the user use floppy disk to transfer the exploit or even manually code the exploit on the victim machine.

- Bug Exploits

Hostile packets might attempt to attack poorly tested software. For example buffer overflow vulnerability are often used to get administrator access or denial out of services (DOS). It is often the case that the services is found in the least used features of the program because otherwise it would be easily identified under normal operation. The input buffer overflow can be seen as an anomalous because the length of the input is different from the normal input.

- Response Anomaly

It is often the case after the attack the victim system generates anomalous response. For example Ping of the Death attack, an attack against older operating system by sending significant size of ICMP packets. It is often the case that the system will reach in unpredictable manner due to the oversize packet. Possible reactions include crashing, freezing and rebooting. Nonetheless, those are all abnormal responses because oversize packet must be rejected. That is why intrusion detection must not only monitor the network traffic but also the system call made by the operating system to detect malicious behavior.

- Bug in the Attack. For example APACHE2 exploit, it sends a request with many http headers against an apache server. Eventually the server will slow down and crash. In the 1999 IDEVAL dataset we have seen that this attack was carried out by sending many request in form of "User-Agent: Sioux". However, there was an unnecessary string of "xxxxxxxxxxxxxxxxxxxx" in the header to perform the attack.

It becomes evident that this attack can be category as an anomalous attack because the attack is different not just to the normal traffic but to the "normal attack". Variation of the attacks (polymorphism) will have impact to the signature detection because it will fail to recognize.

- Evasion

It is often the case that the attacker is trying to hide the attack such attempt could be made by IP fragmentation, overlapping TCP segment that does not match, short TTL , etc. As a result improper IDS could not detect such malicious attack, for example IDS that do not reassembly the fragmentation will believe that the packet is normal.

Having seen there are many type of anomalous events, it is clear that in order to detect novel attacks we must build the model by incorporating many attributes. As a result there are many type of intrusion detections. By and large intrusion detection can be categorized as host-based intrusion detection and network-based intrusion detection. Host based focusing on the internal system i.e system call, where the network based focusing on the external traffics that coming in and out to the system. In this paper we are going to focus all the types of anomaly except response anomaly as we are more interested toward inbound network packets where response anomaly is likely to do with system call and outbound traffics.

### 2.3 Network Anomaly Detection

In contrast to signature detection, anomaly detection has the advantage of being able to detect novel attacks but might suffer higher false positive rates. Anomaly detection uses data mining techniques to classify an attack. However, unlike misuse detection where the learning algorithm is trained over labeled data, anomaly detection builds the models from the normal data. Research in misuse detection often uses various standard data mining algorithms [18]. Unfortunately in most cases the labeled dataset is not always available, partly because it takes huge amount of effort for the experts to label the attack from the real dataset and the network environment is consistently changing. For this reason, many researchers concentrate on building anomaly detection. However, the pure "normal traffic" datasets are rarely available too, in this case often the system made several assumption. If any assumptions fail, the performance of the algorithm will deteriorate.

**Assumption I** : *In the training dataset, only 1-2% contains malicious traffic [30]*

**Assumption II** : *The attack traffic is statistically different to the normal traffic [5, 11]*

One of the issues in the anomaly detection is that there is an unclear boundary between normal and anomalous behavior. For instance if the uncertain behavior is not considered anomalous, then the intrusion activity may

not be detected, on the other hand if the uncertain behavior is considered anomalous, then we might have higher false alarms i.e in the case where there is no intrusion [12].

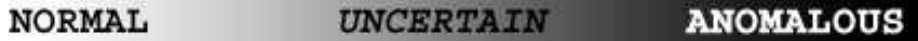


Figure 1: Anomalous behavior must be distinguished from normal behaviour

In general, the difference between various anomaly detections are in the features extracted from the available audit data and the algorithm to build the detection model. Therefore, anomaly detection can be categorized along three dimensions :

- Based on the data being analyzed :packet header or payload
- The level of data being analyzed :packet or application layer
- Probability to classify the attack: Stationary (frequency based model) vs Non-stationary i.e incorporate time

Most anomaly detection systems extract packet header attributes for their model. IP address and port are the most common attributes IDS used. For example for such system is SPADE [10] incorporates four probability models based on the number of frequency:

- $P(\text{source-IP, destination-IP, destination-port})$
- $P(\text{destination-IP, destination-port})$
- $P(\text{source-IP, source-port, destination-IP})$
- a Bayes of network approximation of  $P(\text{source-IP, source-port, destination-IP, destination-port})$

SPADE [10] operates based on the assumption that unseen packets based on IP address and port are likely to be an anomalous, therefore lower probability is higher anomaly score. Another example is ADAM [4]. ADAM [4] is an anomaly detector and classifier than can be trained in both "attack free" model and labeled data. It uses additional header features such as TCP state and subnets. Moreover ADAM [4] uses "Naive-bayes" probability that is it made an assumption that each condition are independent. NIDES [11] likes SPADE and ADAM uses frequency-based model but it differs by flagging between short and long terms model. Obviously, using these limited

attributes to detect novel attack can not be accurate in detecting attacks. There are two reasons for using other attributes : (1) There are many attacks that exploited others than IP and ports, for example attack such as Teardrop's attack crashes the stack by sending overlapping IP fragment, (2) The attacker might evade from the IDS by using various techniques such as fragmentation, bad checksums, short TTL that expired between target and the IDS and so on. Clearly in this case using these limited attributes are not sufficient.

Matt Mahoney et al [21, 22, 25, 19] carried out further research by using many more attributes. PHAD (Packet Header Anomaly Detection) [21] uses Ethernet, IP , TCP , UDP and ICMP packet headers with in total of 34 attributes to build the detection model. The model is universal and there is no distinction being made between incoming and outgoing traffic. Unlike SPADE, ADAM and NIDES use frequency based probability, PHAD incorporates time into calculating the probability of anomalous events. It made an assumption that recent events should predict better, thus giving higher anomaly score based on the last seen anomalous packets. This method has the advantage over non-stationary probability since most attacks tend to occur in bursts separated by time gaps from milliseconds to months [29]. However although PHAD incorporated many attributes in the training scheme, it treats each packet field independently. As a result it cannot detect attacks coming from dependent anomalous event, for example Land attack that crashes SunOS by sending a IP packet that has the same source and destination IP addresses. In this case the fact that source and destination address is the same is an anomalous event but PHAD misses it because it only consider one field at a time.

Network intrusion detection can also be classified on the level of data is being analyzed and modeled. Above systems like PHAD, SPADE, ADAM, NIDES are examples of system that operate on packet level. It has main advantage of being fast because they detect anomalies on the packet level without making an effort to reconstructing it, which is very handy for detecting attacks that do not result in valid connection. The major drawback is that is it can not detect an attack that involve multiple packets within single connection. For example resetscan probes for port by sending RST packet, if there is no response to the reset packet, it means the machine is alive otherwise the router or gateway will respond with "host unreachable" which means the machine does not exist. In this case packet level detection system like PHAD will miss it because it does not maintain the connection thus it won't know that the connection is not open.

It is clear that building the system based on the packet alone is not sufficient. Matt Mahoney carried further research by examining the application

layer of network packet. TCP connections are reassembled from packets based on the TCP options. Application Layer Anomaly Detection (ALAD) [23] applies PHAD techniques probability to TCP streams instead of packet. It is interesting that the best result in 1999 Darpa Dataset was found by incorporating IP addresses, TCP options, Port and the keyword in the payload that is the first word in the connection.[22] shows that combining PHAD and ALAD led to improvement in detection accuracy, however it still has major drawback especially in detecting payload based attack, for example buffer overflow. It is true that in ALAD [23] on 1999 Darpa Dataset, it can detect perfectly Sendmail attack which is SMTP buffer overflow, but it's result is hardly justified because simply the attack was launched from uncommon IP address so instead of detecting from the payload. As a result if the attack was launched from common IP addresss, the system won't be able to detect it. This type of attack hardly appears anomalous,if we only look at the header field, so the only way to detect is by looking directly to the payload.

Several specific payload based intrusion detections exists. For example [37] detects buffer overflow attack via abstract payload execution. Another example would be [16], in this paper the system build the model specifically on web request. Although those systems are be able to detect attacks accurately, it does not provide us with general network characteristics.

To our knowledge ,most of researchers are concentrating building the anomaly detection based on the packet header instead of the payload. The payload based anomaly network intrusion detection is less attractive for two reasons. The first reason is due to encryption; the payload is not generally available in the network until it's being decrypt. So looking for the payload will be useless because the payload will appear uniform. The second reason is due to privacy concern, to our knowledge many datasets that available are not included the payload; it is why it would be very hard to evaluate and compare the performance.

Having said that it does not necessary mean that people are not interested in the payload. The work of Kruegel et al pioneered the work of payload based network anomaly detection in [3]. They use payload as part of the anomaly score and group the ASCII characters in the range from 0 to 256 into six segments that is 0,1-3,4-6,7-11,12-15 and 16-255 and compare the model by using chi-square test. In contrast, Ke Wang et all in PAYL [38] extend the work by using the whole payload as the anomaly score. In PAYL, ASCII characters in the range of 0 to 256 are used to create the distribution without further grouping, therefore it has the advantage of being fast and easy to compute. Furthermore to test the model, PAYL uses simplified mahalanobis distance. Surprisingly payload based detector has a good detection accuracy but might suffer if the data is encrypted.

### 3 Methodology

Our aim is to discover the characteristics of the inbound network payload and use the results to classify future packets. In this experiment we also test our algorithm in both packet and application layers. For application layer it requires some data processing to reassemble TCP connection since the dataset prepared by MIT Lincoln Labs are available in raw TCPDUMP format in per packet model. Furthermore, to simplify our problem we convert these datasets into a well formatted set record for data mining program. So our program only deals only specific attributes set that are needed.

Our work in here is to investigate further anomaly intrusion detection called PAYL [38], by using higher order  $N$ -gram distribution. PAYL is a true payload-based network intrusion detection method that compares payload byte frequency distribution by using Mahalanobis distance.

In general, there are five main challenges to build anomaly network intrusion detection: (1) the intrusion detection should be fast with a low computational cost so it can be deployed in high speed system with little or no impact on latency and throughput; (2) has the ability to cope with changing environment; (3) automatic deployment or requiring no human intervention; (4) resistance to mimicry attack; (5) high accuracy while maintaining low false positive rate.

Furthermore, our focus in this research is on detection and not prevention, i.e. we did not do any immediate steps once we detected the attack, although adding this features should not required any major work but we believe it's beyond our scope.

#### 3.1 N-gram payload byte frequency based length and port

It is true that the attacker tends to exploit the vulnerabilities in the victim that contains private data. Due to the nature of the data in this case the victim tends to employ encryption when sending the data. But we believe that due to the nature of Internet that is open information to public, it is clear that there are many internet servers which do not employ encryption. In this case employing encryption would not have the advantage because the content is meant to be public, on the other hand they still want to keep the server secure for unauthorized access.

By definition payload is just everything after the headers. The content of payload is an ASCII ranging from 0-255. In this case we treat each character as a single ASCII that is "language independent" and build a  $N$ -gram distribution.  $N$ -gram is a probability of an event given all the previous events.

$$P(W_1^n) = P(W_1)P(W_2|W_1)P(W_3|W_1^2)\dots P(W_n|W_1^{n-1}) \quad (1)$$

$$= \sum_{k=1}^n P(W_k|W_1^{k-1}) \quad (2)$$

In this case, we will be using to use N-gram to slice an N-character of a longer string. For example consider the word "TEST" - we will use the underscore character "\_" to represent blanks, therefore the word "TEST" would be composed of the following distributions under various N-grams:

$$1 - \text{grams} : " T " , " E " , " S " , " T " \quad (3)$$

$$2 - \text{grams} : " _T " , " TE " , " ES " , " ST " , " T_ " \quad (4)$$

$$3 - \text{grams} : " _TE " , " TES " , " EST " , " ST_ " , " T_ " \quad (5)$$

$$(6)$$

N-gram has been extensively used in many language analysis tasks as well as security tasks. For example Forrest et al [34] used  $N$ -gram to predict the system calls made by the operating system. Higher order  $N$ -gram such as bigram and trigram were proven to have better accuracy in many language analysis tasks. To illustrate our point consider the word "DOG", using 1-gram it will be composed of ("Character": "Frequency") - (D:1), (O:1), (G:1), but consider the word "GOD", it also composed the same set frequency if we were using 1-gram distribution, therefore the word "DOG" and "GOD" will be match under 1-gram classification but not on higher order n-gram such as bigram and trigram. So it makes sense that higher order N-gram would produce higher accuracy given many training datasets, otherwise we would have many things regard as unclassified.

However for our particular cases we realized that computational and space complexity for computing higher order N-grams such as 2-gram are expensive, therefore the time it takes to compute for higher N-gram is very high. For example for building a 2-gram distribution it takes significantly longer than 1-gram. The reason being that instead of storing the ASCII in the range of 256, 2-gram distribution needs to store the ASCII in the range of 65536 for each network packet thus leaving them 256 times more both computationally and space complexity.

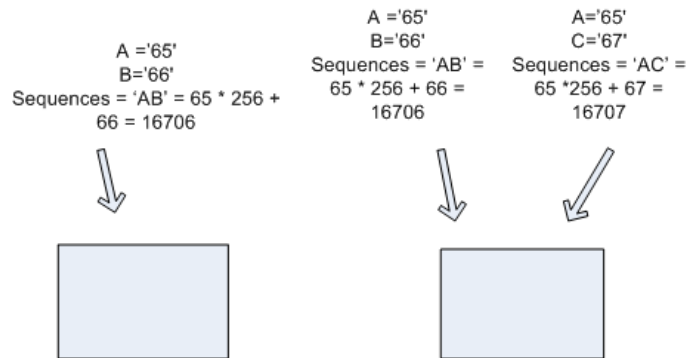
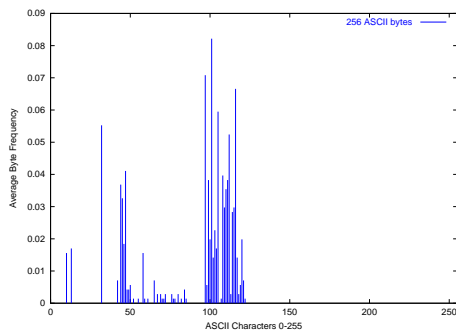
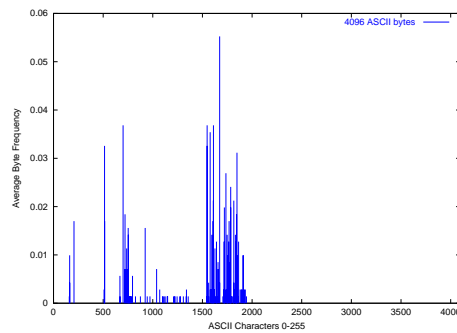


Figure 2: Clustered the sequences 'AB' and 'AC' on the same byte distribution

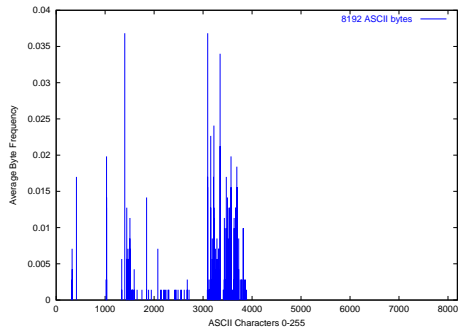
To resolve this problem instead of storing each 2-gram distributions in a single byte, we clustered each 2-gram distribution together, i.e we grouped 2-gram distributions in the range of (0-7),(8-16),(17-24), etc. For instance in the above example (Figure 2) we have the sequences "AB" and "AC" stored on the same byte distribution. Therefore we could reduce both time and space complexities when building the model. Figure 3 shows the different payload distribution under various resolution.



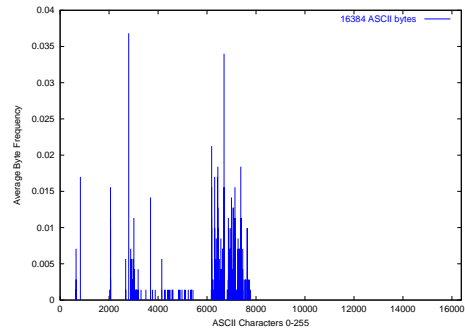
(a) Average Frequency in 256 ASCII bytes



(b) Average Frequency in 4096 ASCII bytes

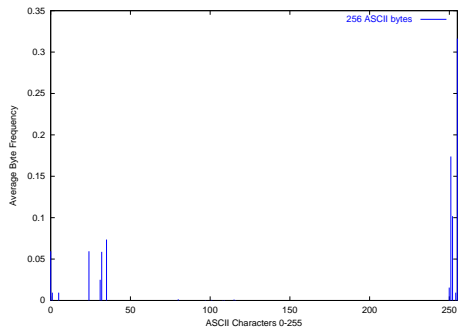


(c) Average Frequency in 8192 ASCII bytes

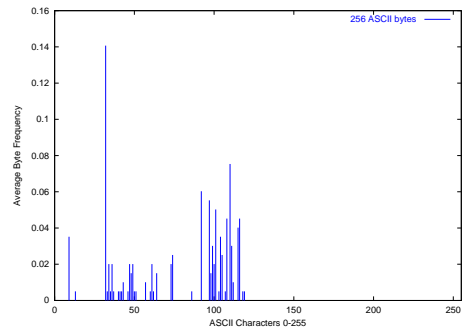


(d) Average Frequency in 16384 ASCII bytes

Figure 3: Example of distribution using 2-gram with various resolution



(a) Average Frequency for length 12



(b) Average Frequency for length 199

Figure 4: Different Length has different distribution

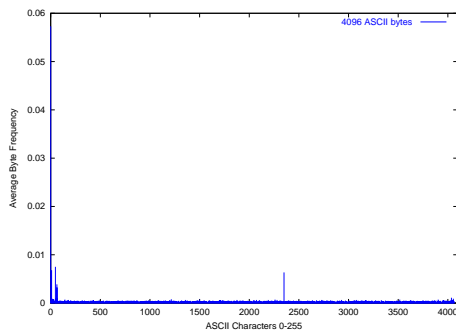
Furthermore, we are not going to have an accurate payload distribution if we build a monolithic model for N-gram payload distribution to cater all length of network packet. Therefore, building the model for specific length is the obvious choices. Various length can be a general indicator for different content, for example a small text file tends to have lower length than a movie file that usually have higher length due to its bigger size. In this case the nature of the payload distribution would be different, where a text file usually contains printable character and movie file contains non printable character and uniform distribution due to compression movie's format (Look at the figure 4 where different length has different payload distribution).

However looking at the length alone to discriminate the content might

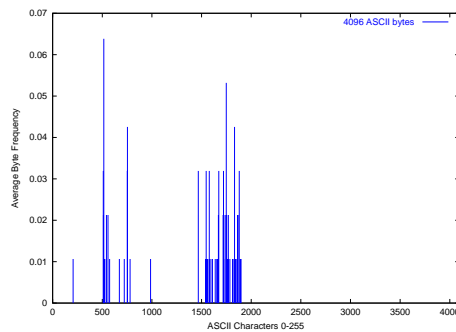
not be appropriate, it is often the case that for a particular length we could have a totally different payload distribution. Consider the above case, where we send a big text file and big video file, often the case the packet is stripped down to the maximum transfer unit. So in this case, the payload of the network packet will both have the same length but different distribution. In this case it makes sense to have multiple models for one particular length, however due to insufficient of training models, we simplify the model so we only build one model for a particular length.

Another obvious choice would be creating the model based on the port number. Each service usually serves at distinct port. For example HTTP service usually serves at port 80, SMTP usually serves at port 25 and SSH usually serves at port 22. Each service has their unique characteristics so in other world their payload distribution would be different. In this case SSH in port 22 their distribution would appear uniformly due to encryption, on the other hand port 23 would appear "free style" due it contains usually are user input where it can be theoretically anything within the keyboard input ASCII.

Furthermore, for each services they also has their own "handshake", for example in port 25 command "HELO" is frequently appear in the payload content where "GET" request is usually appear in HTTP services. So it is make sense to build the distribution based on the port address.



(a) Average Frequency for port 22 (encryption)



(b) Average Frequency for port 23

Figure 5: Different Port has different payload distribution

## 3.2 Mahalanobis Distance

Mahalanobis distance is a standard metric to compare two statistical distributions. It differs from Euclidean distance in a way that it takes into account the correlations of the data set.

The formula of Mahalanobis Distance is shown below :

$$\mathcal{D}^2(x, \bar{y}) = (x - \bar{y})^{-t} C^{-1} (x - \bar{y}) \quad (7)$$

In this equation  $x$  is the feature vector of the new observation where  $\bar{y}$  is the average value from the training dataset.  $C^{-1}$  is the inverse covariance matrix.

Moreover because we made an assumption that each byte is statistically independent. As a result the variance between bytes are zero thus the covariance matrix is just a diagonal matrix with variance for each bytes.

$$\mathcal{D}^2(x, \bar{y}) = \begin{bmatrix} x_1 - \bar{y}_1 & \dots & x_n - \bar{y}_n \end{bmatrix} \begin{bmatrix} \frac{1}{\text{var}(x_1 - y_1)} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\text{var}(x_n - y_n)} \end{bmatrix} \begin{bmatrix} x_1 - \bar{y}_1 \\ \dots \\ x_n - \bar{y}_n \end{bmatrix}$$

The simplified equation is shown below :

$$\mathcal{D}^2(x, \bar{y}) = \sum_{i=0}^{n-1} \frac{|x_i - \bar{y}_i|}{\sigma_i} \quad (8)$$

As one can probably observe, if the covariance matrix is a diagonal matrix then it is the same as normalized euclidean distance, since in this case we take into account the standard deviation. Moreover, there is a chance that standard deviation is zero, thus resulting in false distance. In this case we add small  $\alpha$  coefficient to avoid overfitting.

The modified equation is shown below :

$$\mathcal{D}(x, \bar{y}) = \sum_{i=0}^{n-1} \frac{|x_i - \bar{y}_i|}{\sigma_i + \alpha} \quad (9)$$

The higher  $\alpha$  coefficient indicates lower confidence that the sample dataset truly represent the actual distribution, while lower coefficient is the vice versa. It is make sense that as more samples observed in the training data  $\alpha$  might get decremented.

### 3.3 Incremental Learning

Anomaly network intrusion detection needs to be able to shift or drift to the new environment. Upgrading new services might lead to different payload distribution for example updating the web sites by changing the filename lead to different payload distribution because the user will have different request content as the filename is changed. Ideally intrusion detection should have be able to adapt to such dramatic changed in the environment and at the same time keep maintaining the old model because "history" is an important lesson to predict the future.

One solution is through incremental learning. In this case we update the mean value and standard deviation every time we seen new non-attack packet.

The detail on how to update, we shown below.

$$\bar{x} = \sum_{i=1}^N x_i / N \quad (10)$$

We realized that it is time and space consuming to update the mean every time the new value coming because based on this formula we need to keep previous value to compute the mean. To solve this problem we used clever update by Knuth [15]. As a result we only need to store the current mean value.

$$\bar{x} = \frac{\bar{x} \times N + x_{n+1}}{N + 1} = \bar{x} + \frac{x_{n+1} - \bar{x}}{N + 1} \quad (11)$$

Furthermore, to update the standard deviation, we only need to store the  $x_i^2$  of the model, thus leaving the update very efficient and easy to compute.

$$Var(X) = E(X - EX)^2 = E(X^2) - (EX)^2 \quad (12)$$

$$\sigma^2 = \frac{\sum x^2}{n} - \bar{x}^2 \quad (13)$$

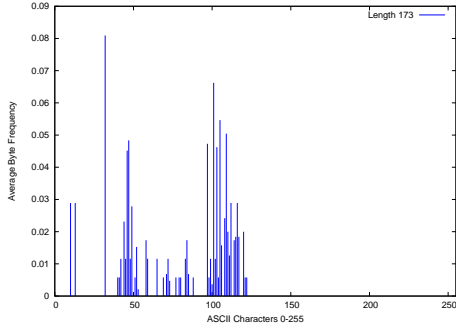
### 3.4 Clustering

It is often the case that on testing phase we are testing a model for a particular length that has not appears in the training dataset. In that case there are two possibilities that can happen: (1) classify them as anomaly because if we could not find the training model for a particular length it could be the case that the packet length is anomalous traffic because otherwise it should

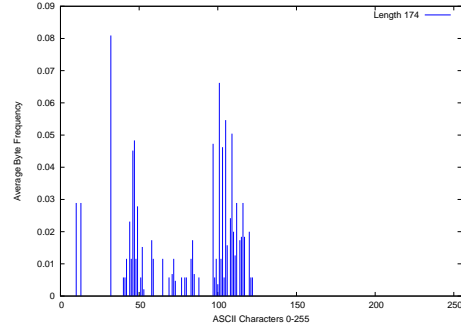
appear somewhere in training dataset (2) find the closest length. The first possibility is true for continues attack. For example APACHE2 attacks that send a lot of "User-Agent: siaoux" request might have different length than normal request. We tried to implement the first solution by giving higher anomaly score to unseen length. Although this technique gives particular attack lower false alarms, overall we had lower detection accuracy. Again, in this case we suspect because there is insufficient of training dataset. So we use the second solution that is finding the closest length.

Furthermore, the length bins  $i-1$  and  $i+1$  might have the same payload distribution because after all it.s only different by few characters. Due to insufficient of training dataset we possess, we join the two length bins distribution together based on their Manhattan distance score that is if the average byte frequency for each distribution is less than the threshold we joined the two distributions together.

$$ManhattanDistance = \sum_{n=0}^N |\bar{x}_i - \bar{x}_j| \quad (14)$$



(a) Average Frequency for Length 173



(b) Average Frequency for Length 174

Figure 6: Length bins 173 and 174 is almost the same

The detail of algorithm is shows below.

---

### Algorithm 1 Clustering Algorithm

---

**Input:** model for various length

**Output:** clustered distribution {clustered}

$x$  = the initial length  $y$  =  $x$ 's neighbored

while ( $y$  is not the maximum length)

If ( $\text{compare}(x,y) < \text{threshold}$ )

then

merge  $x$  and  $y$  + merge any previous length - merge

else  $x = y$ 's neighbored ,  $y = x$ 's neighbored - increment  $x$  and  $y$

---

To illustrate the algorithm that we used we give an example on the figure below. The number represents the mean value for each particular length. In this example we specified that we merges the model together if the different between two particular length is equal or less than 2.

In this case for the first iteration , we compare  $\{4,6\},\{6,8\}$  and  $\{11,15\}$ . The first two tuples the differences is equal to two so refer to our threshold that is if the differences is equal or less than 2 then we merge them together. The next iteration is the same as the previous iteration and so on.



## 4 Evaluation

### 4.1 Evaluation of Darpa Dataset

One of the issues with testing Intrusion Detection is finding the suitable dataset so many researchers can reproduce the result and independently compare the result. Much of the evaluation based on the internal dataset where the dataset can not be shared to other due to privacy concern. Moreover, for the same reason many datasets that are available do not include the payload content. To overcome this problem Lincoln Laboratory under sponsorship of DARPA created the IDEVAL datasets that serve as an evaluation benchmark.

They set up the simulated local area network consists different hosts and internet connection together with simulated attack from published exploit. The datasets contains daily file system dumps, audit logs, network traffic and BSM (Solaris System Call) logs. It consists of 5 weeks dataset. There are two phase on the evaluation, the first phase each participants are given Week 1-3 for building the model for their IDS. Week 1 and Week 3 are attack free and used as a training dataset. Week 2 contained label attack dataset for testing. 6 months later, during the second phase they given week 4 and week 5 datasets that contains label attack for testing. In the 10 days (week4-5) datasets it contains 201 instances of 58 different attacks and about half of them are novel attacks with respect to the first phase.

Moreover, Matt Mahoney in [24] believed that the dataset is not close to the real traffic. He identified that the training dataset contents lesser different attributes than the real traffic dataset they have. In fact it is impossible to simulate the real world traffic [9].

The following is the summary of their findings :

- Packet Header
  - Source Address  
IDEVAL: In week 1 and 3 there are only 29 distinct source IP addresses. All of them appear on the first 0.1% of the traffic.  
Real Traffic: There are 24924 distinct source IP address and 53% of packets seen on the second half of the data. This suggesting that r increases at steady rate. Moreover 45%, of these appear in only a single TCP session, compared to none in IDEVAL.
  - Checksum error  
IDEVAL: There are no IP, TCP, UDP, or ICMP checksums in the 12 million packets found in Week 3.  
Real Traffic : About 0.02% of the traffics has checksums error.

- Time to Leave
  - IDEVAL: There are only 9 distinct TTL values from 256 possible values found in the IDEVAL dataset.
  - Real Traffic: There are 177 different values.
- IP Fragmentation
  - IDEVAL : No IP fragmentation found in the dataset.
  - Real Traffic : About 0.45% of packets are fragmented.
- Payload Content
  - HTTP Request
    - IDEVAL: In 16,089 HTTP request they always have in the form on .GET url HTTP/1.0. followed by option syntax in the form of .Keyword: values.. Moreover, the first letter of the keyword is always upper case with a space after the colon. They only have 5 distinct versions in the User-Agent field that is all versions of Mozilla, Netscape and Internet Explorer.
    - Real Traffic: In 82,013 real HTTP request although 99% of the request is GET, but there are also other request seen such as HEAD, POST, OPTIONS, PROPFIND, LINK and two malformed commands "No", and "tcp\_close". The keyword is not always capitalized and the spacing around the colon is not consistent. In the real traffic is also found that some of the keywords are misspelled and malformed. In contrast, in the real traffic they have 807 different user agents. The top five are : "Scooter3.2", "googlebot/2.1", "ia\_archiver", "Mozilla/3.01" and "http://www.almaden.ibm.com/cs/crawler". In which, 44% of the agent appear only once in the dataset.
  - SMTP Request
    - IDEVAL: In 18,241 SMTP mail request, HELO or EHLO (echo hello) command always initiated the request. There are 3 distinct HELO arguments and 24 different EHLO arguments found, which all but one which appear at least twice.
    - Real Dataset: In 12,911 SMTP mail request, there are 1839 distinct HELO arguments in which 69% appearing only once and 1461 different EHLO arguments in which 58% appeared only once. Moreover, 3% of SMTP requests do not start with HELO and EHLO commands, instead they start with RSET, QUIT, EXPN, NOOP or CONNECT. It's also important to note that 0.05% of the real request used lower case and 0.1% of the requests are malformed but none in the IDEVAL.

It suggests that IDEVAL datasets are more predictable than the real dataset. For this reason it is likely the case that anomaly detection rule will generate more false alarms in the real dataset than in the IDEVAL dataset. For example, Matt Mahoney et al suggests that the used of TTL field in PHAD [21] to detect the attack might misleading the detection. As previously discussed there are only 9 distinct TTL values (2, 32, 60, 62-64,127-128,254-255) in the training dataset, where most detections and false alarm in the testing dataset are due to TTL result from the anomalous values 126 or 253, which absent in the training dataset. That is without having TTL field; PHAD can only detect 48 attack instances, instead of 67 instances.

Furthermore, one of the disadvantage of the dataset is that the simulation of the attack is perform without further enhancing the technique that is the way they perform the attack is by downloading the script from the security web site and run it to the victim. Although the attacker who perform the simulation might understand the great detail of the attack but these kind of techniques that's not involving the creativity of the attacker is often used by "script kiddies" , that is low level of hacker. So in this case there is no attempt to make the attacks appear stealth.

Eight organizations submitted 18 intrusion detection systems [33]. An attack is detected if the IDS can generate the correct victim IP address and the time of any potion of the attacks within 60 seconds. The IDS also need to generate the anomaly score for each alarm. Duplicate detection of the same attack is counted once by picking the highest anomaly score within the time interval, but every false alarm is counted. The best 4 are shown below at the rate of 1% false positive.

System	Detection
Expert 1	85/169(50%)
Expert 2	81/173(47%)
Domine	41/102(40%)
Forensics	15/27(55%)

Table 1: Top 4 result of 1999 Darpa IDS evaluation

## 4.2 Experimental Setup

The experiment was carried out on student server IBM eServer P650 series. It powered by eight of 1.45 GHZ CPUs and 16 GB of memory.

For this experiment we use Week 1 (5 days) and Week 3 (7 days) which are attack free as a training model. These datasets consist of 2.3 GB of

training datasets. Week 4 and Week 5 which contains the labeled attacks we used in the experiment as testing dataset to measure our proposed system.

In this experiment we only focused on TCP so any other attack coming from other than TCP such as UDP, ICMP, ARP and IP only can't be detected. Furthermore, Week 4 and Week 5 consist of 201 labeled attack instances. However, many of the attacks can not be detected through the payload. For example Land attack, the anomaly from this attack can only be identified by looking at the header field that is the source is the same as the destination address. Another example would be Ping of the death (POD) attack where the attack where they can only be identified by looking at oversize of ICMP packet. So in this case it would be reasonable to remove these kinds of attacks because it's not related to our system. In fact including them in our detection scenario might distort the accuracy of the result.

We also focused on 172.16.xxx.xxx destination IP address so attacks that go to other than this IP range will be deleted. Moreover the dataset for Week 4 . Tuesday isn't available. So after filtered based on these attributes there are 98 attacks on the dataset that we used for experiment. That's also including unlabeled attack that we accidentally found by looking at the payload.

To further speed up the experiment, we also did filtering on the raw dataset. The idea so we can get the result in faster and at the same time we could focus on the algorithm performance because in most cases handling with huge size of datasets could lead to I/O bound problem that is the CPU waiting for the I/O to finish read the dataset. The result is for the training dataset we could reduce the size to 128 MB instead of reading the file from 2.4GB. Please note that in this case we also remove unnecessary header.

Moreover we initially believe the way we should conducted the experiment is by specifying the threshold for the anomaly. What we meant by that is that if the anomaly scores above some specific threshold then we generate the alarm. This technique has the disadvantage of hardly limit the false positive rate, so what we did instead is we printed all the anomaly score, time, destination IP address and sorted in ascending order. At the same time we also give 60 seconds leeway period for each attack. In fact this technique is the same technique that 8 contestants used in 1999 Darpa Evaluation [33].

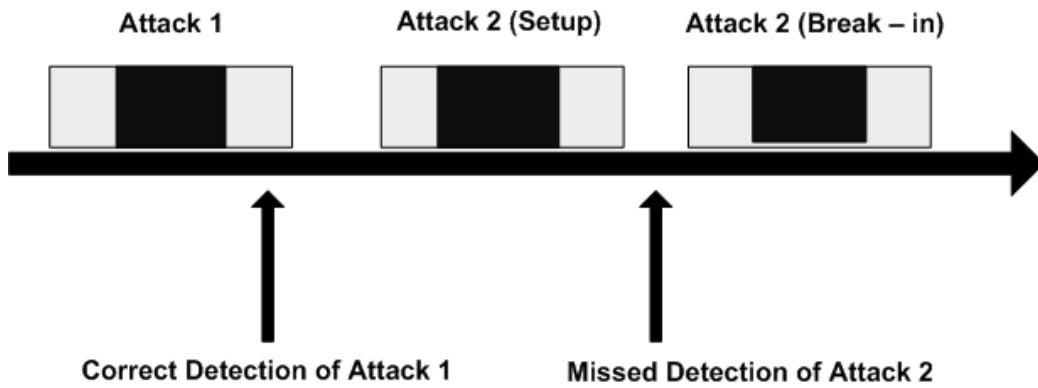


Figure 8: 60 seconds leeway period for each attack

One disadvantage of using these attributes that is time, destination IP address and anomaly score is that it can be the case the intrusion detection claimed they detected the APACHE2 attack based on the anomaly score intended to the same IP address but different port, so for example it is possible that it generated alarm from other than port 80 packets, which we know that APACHE2 attack is intended for Apache daemon that located in port 80. Since we building the model based on port and length of payload, it is reasonable to straightened the evaluation by adding the port number as evaluation attributes.

### 4.3 Result

We plotted the detection accuracy for major ports such as 21,23,25 and 80. We can evaluate the performance by limiting false positive rate to obtain different detection accuracy. Although it is true that there are some attack that intended other than these port such as Dosnuke attack, an attack that intended to port 139, the result from this port is not meaningful because it lacks of training dataset.

For comparison, we tried to use different number of resolution of 2-gram. In this particular case we used 4096, 8192 and 16384 numbers of resolution. 4096 means that 65536 of 2-gram distributions are store in the 4096 bytes by grouping them. In addition to this, we also used three different data units, for both training and testing :

- Per Connection model, which uses the whole payload for each connection
- Per Connection truncated, which uses the first 1000 bytes of each connection
- Per Packet model, which uses the whole payload for each network packet

To more accurately compare these approaches, we plotted the performance in terms of speed of each data units and N numbers of resolution. In this case, we tried to build 1000 data units for each model.

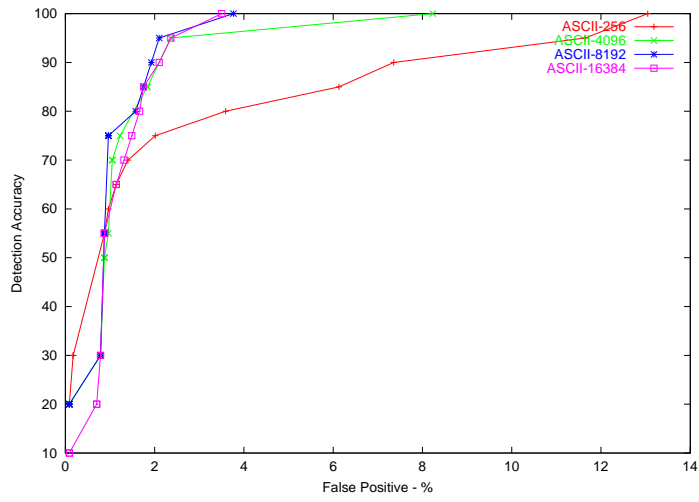


Figure 9: Port 21 Connection

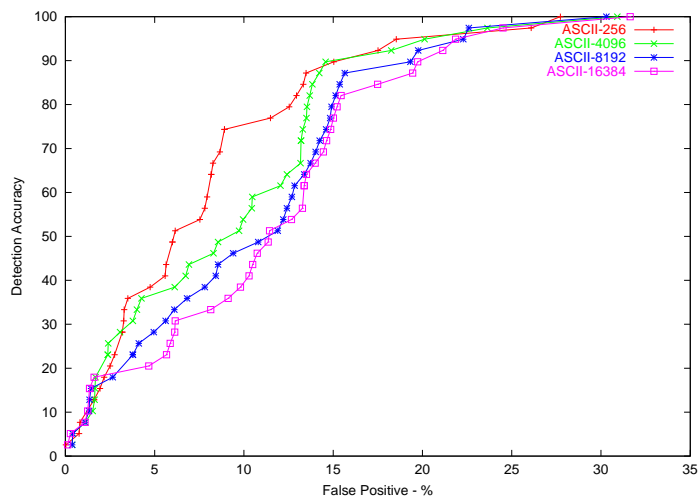


Figure 10: Port 23 Connection

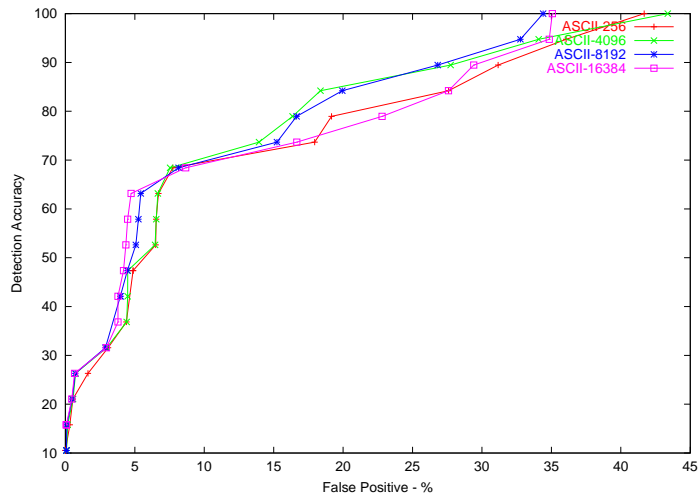


Figure 11: Port 25 Connection

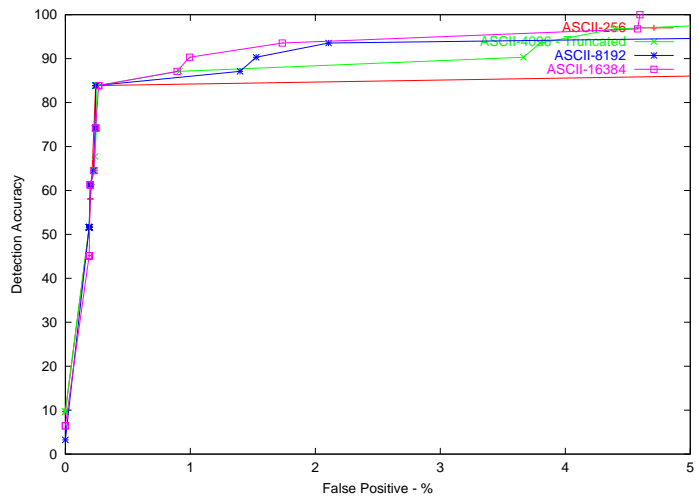


Figure 12: Port 80 Connection

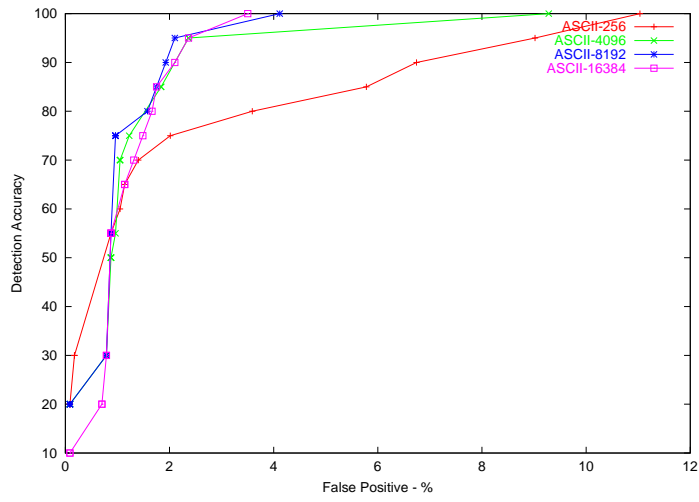


Figure 13: Port 21 Connection truncated

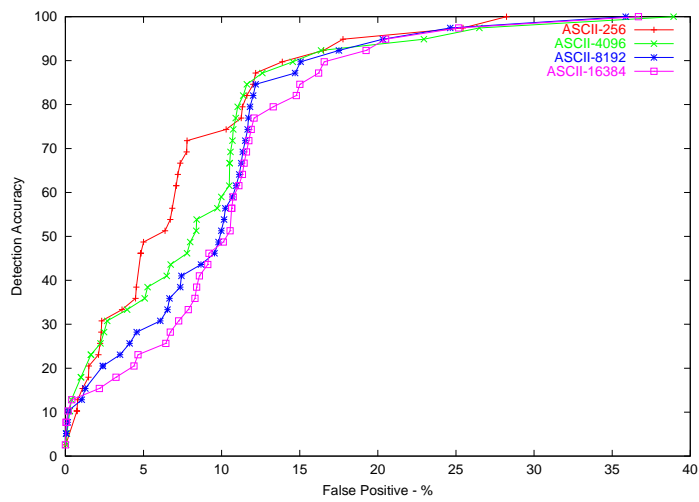


Figure 14: Port 23 Connection truncated

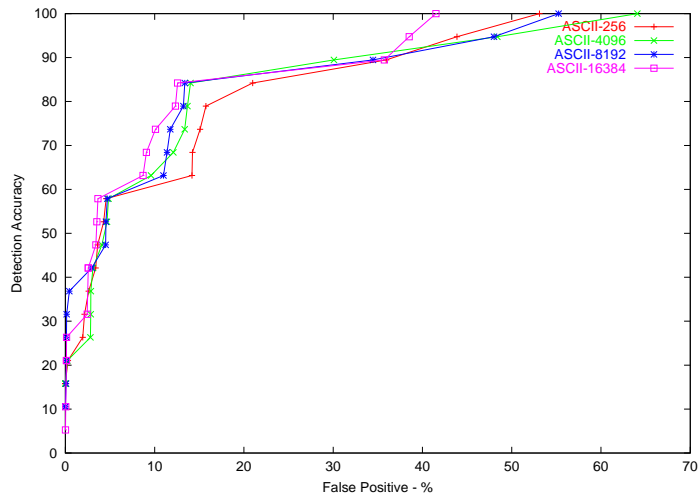


Figure 15: Port 25 Connection truncated

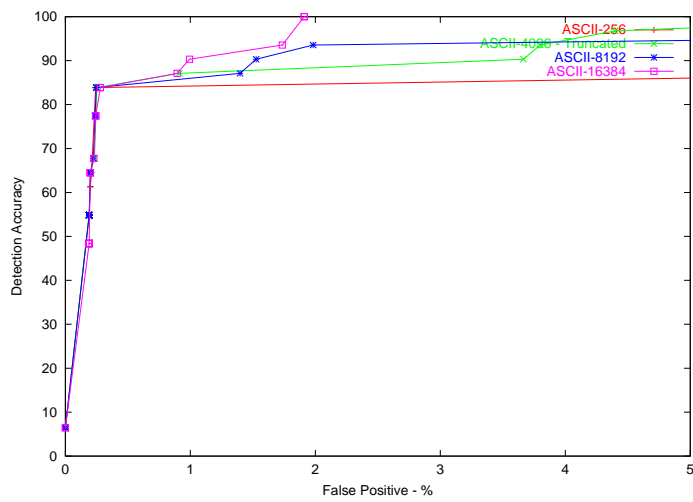


Figure 16: Port 80 Connection truncated

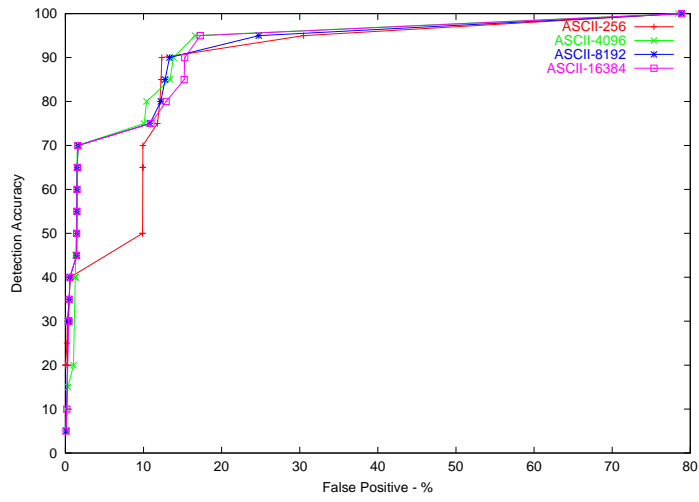


Figure 17: Port 21 Packet

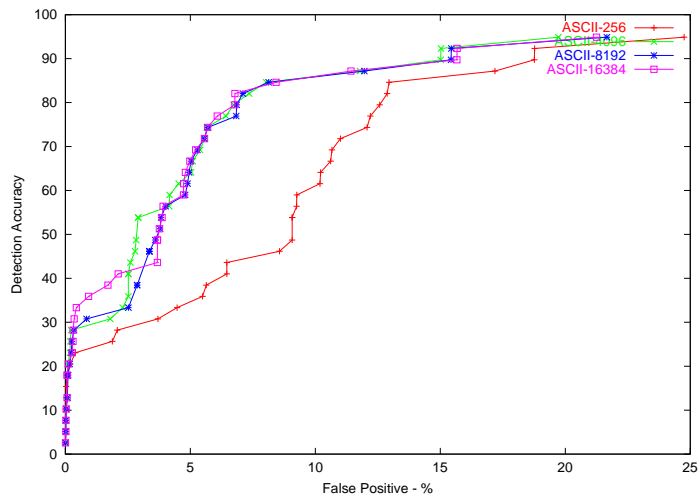


Figure 18: Port 23 Packet

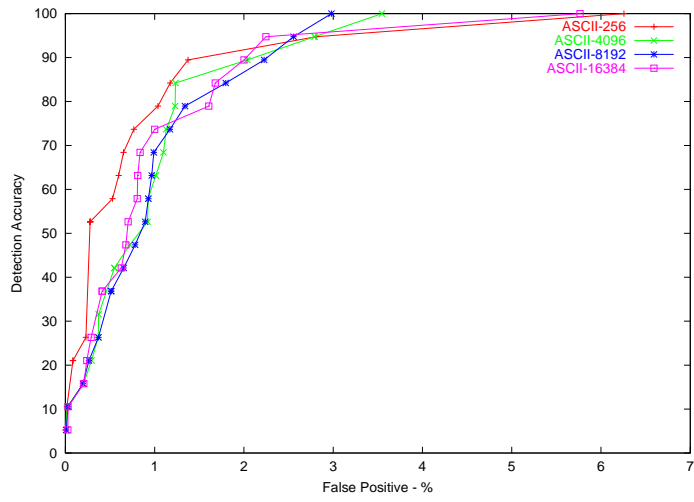


Figure 19: Port 25 Packet

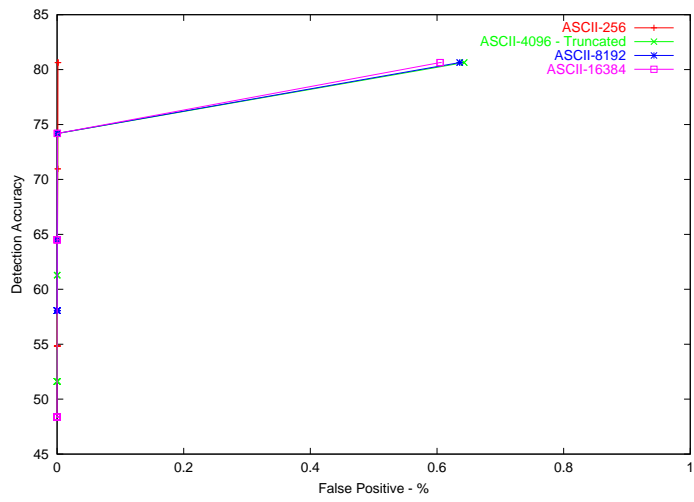


Figure 20: Port 80 Packet

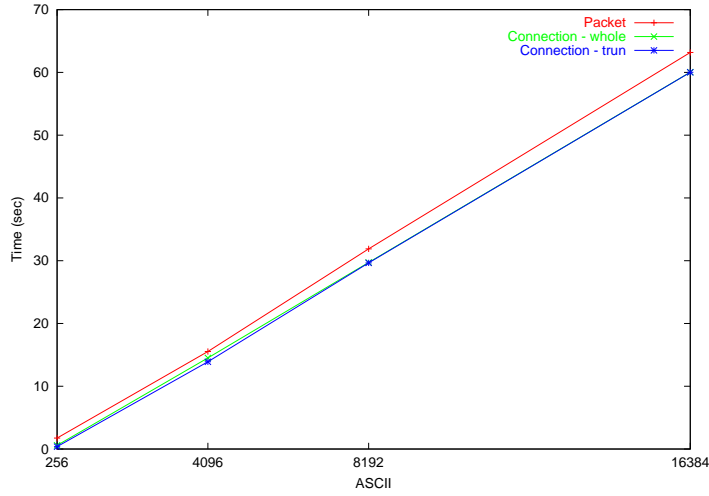


Figure 21: Performance Graph for various N-gram resolution

ASCII bytes	Per Connection	Truncated Connection	Per Packet
256	0.580	0.370	1.75
4096	14.510	13.9	15.540
8192	29.710	29.650	31.890
16384	60.0320	60.020	63.179

Table 2: Time taken to build the model in seconds

There are many attacks that involve multiple steps by exploiting multiples ports. For instance ffbconfig, in this dataset an attacker who is exploiting the service must first transfer the exploit code into the victim machine from ftp services (port 21) and execute the code from telnet services (port 23). If we can detect the attack at any one port, then we can detect the attack successfully. We correlate the detector from all the major port and plot the overall performance.

ASCII bytes	Per Connection	Truncated Connection	Per Packet
256	40/85(47%)	45/85(53%)	54/85(63%)
4096	40/85(47%)	51/85(60%)	46/85(54%)
8192	43/85(51%)	55/85(65%)	54/85(63%)
16384	42/85(49%)	53/85(62%)	54/85(63%)

Table 3: Overall detection

## 5 Discussion

### 5.1 Result Discussion

As we can see, payload based model is very good in detecting attack that are intended to port 80 or Web Server services. Partly the reason is because most of the attacks in port 80 are very different from normal request. For example, Back attack where it's payload has substantial number of front slashes, in this case 100 front slashes, which are very different from the normal request. The same case applied for Apache2 attack where it sends a lot of repeated "User-Agent:sioux". In this case using 1-gram is good enough in detecting those attack. However, further inspection of the dataset revealed that for an attack that is not very different from the normal request, having an 2-gram distribution gives an advantage. For instances ppmacro where the attacker download the infected ".ppt" file from the web, it turns out that using 2-gram could detect this attack in lesser false positive rate. One explanation for this is because extension ".ppt" appears less frequent in the training dataset.

Moreover, payload detector are also good in detecting attacks in port 21 or FTP's service. For instance warezmaster and warezclient which are attacks that involved upload/download illegal copies of software, where the content of the warez file is quite different to the normal traffic. Careful inspection to the test dataset, using higher N-gram distribution to detect the attack targeted to this port has proven to be advantageous, for instance guessftp attack can be detected in less false positive rate due to the attack tried to use the same username and password where in the normal traffic it is unlikely the case the user have the same password as the username.

Most of the attack aimed to port 23 are U2R(User to Root) type of attack. It means that the user have legitimate access to the system and executed the exploit from the victim system. In this case, payload based detector hardly recognized the attack since in the network point of view this would appears as a normal behavior. Looking at the testing data in detail revealed that for some attack it leaves a distinct signature, for instance eject attack in this dataset it printed "Jumping to memory", but this could easily removed from the source code thus make the attack is well hidden. More over we found that using higher order N-gram for per packet model in port 23 does not have added advantage. The reason is because 80 % of the packet traffic that goes to port 23 only contains single character or packet length equal to one. This phenomena also has an impact to the detection accuracy making them less reliable. For instance the system could claim a particular attack being detected based on a single character, where it might be the case for that particular character rarely appear in the dataset.

Port 25 also shared the same problem as port 23 because the content is quite random except the protocol. For instance attack such as sendmail where the protocol is quite different since the attack sends a MIME header line that is inappropriately large therefore it could easily be detected by either 1-gram and 2-gram payload detector. However for attacks like mailbomb where thousands of mail messages sent to a particular user, it would be classified as normal traffic due to the random nature of the content. Although we found that 2-gram has lesser false positive rate in detecting this attack, we believe that this is unjustified because rather than making an association to the number of messages sent within a small time period, payload based detector judge the attack based on the email's content which in this case it happened to be anomalous. This introduces problem as it will generate false alarms easily, for instance in English we have the character "e" as the most common character. It becomes clear that every language as a unique property therefore it is possible for someone sent an email in other language could generate a false alarm.

It is interesting that 2-gram distribution with lower resolution is not necessary better in terms of detection accuracy. In this case, the overall performance shows that 8192 ASCII bytes is better than 16384 ASCII bytes. The reason being that we do not have sufficient training dataset therefore unseen sequences character will have higher distance. This phenomena is common in many natural language problems or it refers to data sparse problem. We believe that large training dataset could lead improve in accuracy, the reason for this is unlike natural language problem where the dataset is quite random, the network payload is quite static because it involved some of the protocol messages, therefore it's quite predictable especially for port 21 and 80. For instance HTTP request field are explained in the RFC822 format headers, so the structure are clearly defined. Although this is not true for port 23 as described above.

## 5.2 Performance Discussion

Performance study using 2-gram revealed that the time takes to build the model is linear. It means that if computing the model of 1-gram that is 256 ASCII bytes take 1 sec, thus to build complete distribution of 2-gram takes 256 times complexity i.e 256 sec. It is important to note that although in the table, the time it takes to build the per packet model is higher than per connection model, in reality the reverse is true. The reason why per packet model have longer time is because for per connection model we have pre-processed the dataset, so the reconstruction of the payload is not done in the either training or testing phase. On the other hand it reveals that

the time taken depend on the number of packets rather than the length of the packet, because the performance for packet and connection model are similar, although obviously the payload length for 1000 connections are higher than 1000 packets. This has implication on deploying on the real time environment. One of the issues of deploying IDS in the real time environment is packet drop. In this case our system has resistant dealing with higher number of packet rather than the size of the packet.

## 6 Future Work

Due to the lack of time there are still areas that were unable to be investigated and improved on the proposed mechanism. In this section we suggest some areas of the project as topics of possible future work.

- Higher order N-gram introduces sparse data problem. There are many approaches to smooth the probability function to cope with the sparse data problems. One of the them is to use some sort of back-off or interpolated estimator and check whether the result improved.
- The idea of using MIT Darpa dataset so we can compare our methods with others and also so that other researchers can verify our result. One of the issues in the payload based intrusion detection is it is very hard to find the dataset that is available for public. Many of them are available for private consumption due to privacy concern. So it is reasonable to say that one of the future tasks is to try to use our algorithm in real life dataset. For this thesis we regret that we could not have a chance to test our algorithm with such dataset because the process to gather the data is time consuming.
- The idea of using time component in the detection is to be able to detect burst attack. In this topic our focus is only to detection but in the future we might extend the focus into prevention. This was based on our believe that DDOS (Distributed Denial out of Service) attacks like SMURF attack send the same ping packet from vulnerable clients to the victim. Moreover it's often the case especially with worm attacks that the attack packet coming in to the victim is the same as output packet coming from the infected machine. In this case looking at the payload distribution has a good incentive.
- Due to heterogeneous nature of the internet it is believe that we should use as many as attributes to build the model in order to detect various attacks. In this thesis we only build the model based on the payload so there are many non-overlap techniques that could be use, for example PHAD [21] is one of them. We shall further investigate the effectiveness of detecting the attack by combining these two techniques.
- As previously discussed for a particular length and port, they could also have different payload distribution. For example PDF file might have different distribution to Text file. In that case in order to improve the accuracy we might have multiple models for particular length. Simple

or advance clustering technique could be used to group the similar distribution for the same length. Having said that to apply this method we also need to find large datasets in order to build the model correctly.

## 7 Conclusion

We have evaluated payload based network intrusion detection called PAYL [38] and checked whether incorporating higher N-gram would have an impact to the detection accuracy. In this case we employed 2-gram with various resolution numbers. Our experiment suggests that in general 2-gram distribution has increased the detection accuracy, but lower resolution does not guarantee to lead in higher accuracy. In this case using 8192 bytes to store the 2-grams distribution proven to be the optimal resolution. Performance study also reveals that the time taken of having higher order N-gram is linear to the number of ASCII bytes. Therefore one of the tradeoff of using higher order N-gram is higher computational time.

## References

- [1] Manasi Bhattacharyya, Matthew G. Schultz, Eleazar Eskin, Shlomo Hershkop, and Salvatore J. Stolfo. Met: An experimental system for malicious email tracking. In *3rd Annual IEEE Information Assurance Whorkshop*, 2002.
- [2] CERT Coordination Center. <http://www.cert.org>.
- [3] Engin Kirda Christopher Kruegel, Thomas Toth. Service specific anomaly detection for network intrusion detection. *SAC*, 2002.
- [4] Barbara D., N. Wu, and S. Jajodia. Detecting novel network intrusions using bayes estimators. In *First SIAM International Conference on Data Mining*, 2001.
- [5] D.E. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering*, 1987.
- [6] eEye Vulnerability Assesment and Intrusion Prevention Network Security Software. <http://www.eeye.com>.
- [7] Juan M. Estevez-Tapiador, Podro Garefa-Toodoro, and Jesus E. Dfaz-Verdejo. N3: A geometrical approach for network intrusion detection at the application layer. *ICCSA*, pages 841–850, 2004.
- [8] Dan Farmer and Wietse Venema. Improving the security of your site by breaking into it. <http://www.fish.com/security/admin-guide-to-cracking.html>.

- [9] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 2001.
- [10] J. Hoagland. Spade, silicon defence, 2000. <http://www.silicondefense.com/software/spice>.
- [11] H.S. Javits and A. Valdes. The nides statistical component: Description and justification. Technical report, SRI International, Computer Science Laboratory, 1993.
- [12] Anita Jones and Robert Sielken. Computer system intrusion detection: A survey. Technical report, University of Virginia, 2000.
- [13] Gabriela Cretu Ke Wang and Salvatore J. Stolfo. Anomalous payload-based worm detection and signature generation. *Usenix Security*, 2005.
- [14] Kris Kendall. *A Database of Computer Attacks for the Evaluation of Intrusion Detection*. PhD thesis, Massachusetts Institute of Technology, Boston, NY, 1998.
- [15] D.E Knugh. *the Art of Computer Programming, Vol 1 Fundamental Algorithms*. Addison Wesley, 1973.
- [16] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. *ACM*, 2003.
- [17] Wenke Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, New York, NY, 1999.
- [18] Wenke Lee and Salvator J. Stolfo. A data mining framework for building intrusion detection models. *IEEE Symposium on Security and Privacy*, 1999.
- [19] Matt Mahoney. *A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic*. PhD thesis, Florida Tech, Melbourne, Florida, 2003.
- [20] Matthew V. Mahoney. Network traffic anomaly detection based on packet bytes. In *ACM*, pages 346–350, 2003.
- [21] Matthew V. Mahoney and Phillip K. Chan. Phad: Packet header anomaly detection for identifying hostile network traffic. Technical report, Florida Institute of Technology, 2001.

- [22] Matthew V. Mahoney and Phillip K. Chan. Learning models of network traffic detecting novel attacks. Technical report, Florida Institute of Technology, 2002.
- [23] Matthew V. Mahoney and Phillip K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Eighth Intl Conf. Knowledge Discovery and Data Mining*, pages 376–385, 2002.
- [24] Matthew V. Mahoney and Phillip K. Chan. An analysis of 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. Technical report, Florida Institute of Technology, 2003.
- [25] Matthew V. Mahoney and Phillip K. Chan. Learning rules for anomaly detection of hostile network traffic. In *Third International Conference on Data Mining*, 2003.
- [26] John McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluation as performed by lincoln laboratory. *ACM Transaction on Information and System Security*, 3:262–294, 2000.
- [27] Darren Mutzi and Christopher Kruegel. Reverse engineering of network signatures. In *AusCert*, 2005.
- [28] V. Paxson. Bro: A system for detecting network intruders in real-time. *Usenix Security*, 1998.
- [29] Vern Paxson and Sally Floyd. The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 1995.
- [30] Leonid Pornoy, Eleazar Eskin, and Salvatore Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security*, 2001.
- [31] Nessus Open Source Vulnerability Project. <http://www.nessus.org>.
- [32] David J. Fried Richard Lippman, Joshua W. Haines and Kumar Das. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. Technical report, MIT Lincoln Library, 2000.
- [33] David J. Fried Richard Lippmann, Joshua W. Haines and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. Technical report, MIT Lincoln Laboratory, 2002.

- [34] Forrest S., S.A Hofmeyr, A. Somayaji, and T.A Longstaff. A sense of self for unix. *IEEE Symposium on Computer Security and Privacy*, 1996.
- [35] Salvatore J. Stolvo and Ke Wang. Fileprint analysis for malware detection. Technical report, Columbia IDS Lab, 2005.
- [36] Snort the de facto standard for intrusion detection/prevention. <http://www.snorg.org>.
- [37] Thomas Toth and Christopher Kruegel. Accurate buffer overflow detection via abstract payload execution. Technical report, Technical University of Vienna Information Systems Institute, 2003.
- [38] Ke Wang and Salvatore J. Stolvo. Anomalous payload based network intrusion detection. *RAID Security*, 2004.