Spring 2013

HTTP Attack Detection using N-gram Analysis

Adityaram Oza
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Oza, Adityaram, "HTTP Attack Detection using N-gram Analysis" (2013). *Master's Projects*. 299.
DOI: <https://doi.org/10.31979/etd.rbtj-p2jh>
https://scholarworks.sjsu.edu/etd_projects/299

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

HTTP Attack Detection using N -gram Analysis

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Adityaram Oza

May 2013

© 2013

Adityaram Oza

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

HTTP Attack Detection using N -gram Analysis

by

Adityaram Oza

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2013

Dr. Mark Stamp Department of Computer Science

Dr. Robert Chun Department of Computer Science

Dr. Richard Low Department of Mathematics

ABSTRACT

HTTP Attack Detection using N -gram Analysis

by Adityaram Oza

Previous research has shown that byte level analysis of HTTP traffic offers a practical solution to the problem of network intrusion detection and traffic analysis. Such an approach does not require any knowledge of applications running on web servers or any pre-processing of incoming data. In this project, we apply three n -gram based techniques to the problem of HTTP attack detection. The goal of such techniques is to provide a first line of defense by filtering out the vast majority of benign HTTP traffic. We analyze our techniques in terms of accuracy of attack detection and performance. We show that our techniques provide more accurate detecting and are more efficient in comparison to a previously analyzed HMM-based technique.

ACKNOWLEDGMENTS

I want to thank Dr. Mark Stamp for introducing me to the fascinating science of information security and for his encouragement and guidance throughout my graduate studies.

I would also like to thank Dr. Chun and Dr. Low for agreeing to be on my project committee and for their valuable inputs throughout the course of the project.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Background	3
2.1	Network intrusion detection systems	3
2.2	Previous research	3
2.2.1	HMM-based detection	4
2.2.2	N -gram based detection	7
2.2.3	Other related work	9
2.3	Pearson's χ^2 test	9
2.3.1	χ^2 Statistic	10
3	Design Overview	13
3.1	χ^2 distance	13
3.2	Ad-hoc n -gram distance	16
3.3	Pattern Counting	16
4	Datasets	19
4.1	Publicly available Datasets	19
4.1.1	Benign Traffic	19
4.1.2	Attack Traffic	20
4.2	Simulated traffic datasets	21
4.2.1	Benign traffic	21
4.2.2	Attack Traffic	22

5	Experiments	27
5.1	Experimental Setup	27
5.2	Performance Evaluation	28
5.3	Experiments on Datasets of simulated HTTP traffic	29
6	Results	30
6.1	Generic Attacks Dataset	30
6.2	Shellcode Attacks Dataset	36
6.3	CLET Dataset	40
6.4	Performance analysis	43
6.5	Results for experiments on dataset of simulated HTTP traffic . . .	46
6.5.1	Experimental Setup	46
6.5.2	Results	46
7	Conclusion and Future work	59

LIST OF TABLES

1	DARPA Dataset	20
2	Attack Datasets	21
3	Simulated Benign Traffic	22
4	Simulated Attack Traffic	24
5	χ^2 distance - Generic Attacks	32
6	Adhoc n -gram distance - Generic Attacks	33
7	Pattern Counting - Generic Attacks	35
8	χ^2 distance - Shellcode Attacks	37
9	Adhoc n -gram distance - Shellcode Attacks	38
10	Pattern Counting - Shellcode Attacks	39
11	Results from HMMPayl - Shellcode Attacks	39
12	χ^2 distance - CLET Dataset	40
13	Adhoc n gram distance - CLET Dataset	42
14	Pattern Counting - CLET Dataset	43
15	Results from HMMPayl - CLET Dataset	43
16	Summary of performance data (Time in milliseconds)	43
17	Number of unique n grams	45
18	Summary of performance data - HMMPayl (Time in milliseconds) . .	45
19	System Configuration	45
20	χ^2 distance - Generic Attacks	47
21	Adhoc n -gram distance - Generic Attacks	48

22	Pattern Counting - Generic Attacks	49
23	χ^2 distance - Php based Attacks	50
24	Adhoc n -gram distance - Php based Attacks	51
25	Pattern Counting - Php based Attacks	52
26	χ^2 distance - MS Buffer overflow Attacks	53
27	Adhoc n -gram distance - MS Buffer overflow Attacks	54
28	Pattern Counting - MS Buffer overflow Attacks	55
29	χ^2 distance - Twiki based Attacks	56
30	Adhoc n -gram distance - Twiki based Attacks	57
31	Pattern Counting - Twiki based Attacks	58

LIST OF FIGURES

1	HMMPayl IDS Architecture	5
2	HMMPayl feature extraction scheme	6
3	Packet scored against multiple HMMs	6
4	File type classification scheme	8
5	χ^2 Distance	15
6	Pattern Counting	18
7	Benign HTTP Request	25
8	Sample Attack Request	26
9	BOX plot - Generic Attacks - χ^2 Distance- 3 grams	31
10	ROC - Generic Attacks - χ^2 Distance	31
11	BOX plot - Generic Attacks- Adhoc n -gram Distance - 3 grams	32
12	ROC - Generic Attacks - Adhoc n -gram distance	33
13	BOX plot - Generic Attacks- Pattern Counting - 4 grams	34
14	ROC - Generic Attacks - Pattern Counting	34
15	Generic Attacks - AUC_p values for n -gram sizes 1–5	35
16	HMMPayl - Generic Attacks - AUC_p values for n -gram sizes 2–10	36
17	ROC - Shellcode Attacks - χ^2 distance	37
18	ROC - Shellcode Attacks - Adhoc n -gram distance	38
19	ROC - Shellcode Attacks - Pattern Counting	39
20	ROC - CLET Dataset - χ^2 distance	41
21	ROC - CLET Dataset - Adhoc n -gram distance	41

22	ROC - CLET Dataset - Pattern Counting	42
23	Performance - per packet processing time (milliseconds)	44
24	ROC - Generic Attacks - χ^2 Distance	47
25	ROC - Generic Attacks - Adhoc n -gram Distance	48
26	ROC - Generic Attacks - Pattern Counting	49
27	ROC - Php based Attacks - χ^2 Distance	50
28	ROC - Php based Attacks - Adhoc n -gram Distance	51
29	ROC - Php based Attacks - Pattern Counting	52
30	ROC - MS Buffer overflow Attacks - χ^2 Distance	53
31	ROC - MS Buffer overflow Attacks - Adhoc n -gram Distance	54
32	ROC - MS Buffer overflow Attacks - Pattern Counting	55
33	ROC - Twiki based attacks - χ^2 Distance	56
34	ROC - Twiki based attacks - Adhoc n -gram Distance	57
35	ROC - Twiki based attacks - Pattern Counting	58

CHAPTER 1

Introduction

Security of large-scale web applications is an important topic in information security. According to [21], approximately 4500 new web based attacks were introduced each day in 2011. According to [4], web application servers are common targets for malware attacks, in part because they communicate with many other systems and therefore help to spread malware.

Open source examples of network intrusion detection systems include [5] and [19].

Previous research has shown that raw bytes of network traffic on the wire can be directly used for intrusion detection and traffic analysis [2, 15, 25]. Such an approach does not require any knowledge of applications running on the web servers or any pre-processing of incoming data.

Most of HTTP traffic for online shopping malls, government websites consists of only text data, i.e., ASCII printable characters. In particular, we do not expect any executable code in incoming HTTP packets. Such executable code can be a possible malware injection attack [1]. Therefore, analysis of the distributions of bytes in HTTP packets can convey useful information about expected HTTP traffic and can help us to detect attacks.

Techniques based on hidden Markov models and n -gram analysis have been used to successfully detect HTTP attacks [2, 15, 25]. In [25] an n -gram model is applied to raw bytes in HTTP packets. The paper [2] considers the use of hidden Markov models to analyze the structure of HTTP packets. The n -gram based technique in [25] has been applied to the problem of file-type classification in [1], where byte

distributions are used to classify files as executable, text, or multimedia. The paper [1] also proposes a pattern counting technique to classify text and executable files. The paper [22] proposes a technique based on a χ^2 statistic and HMMs to detect malware.

In this paper, we employ n -gram analysis similar to that in [1, 25]. We also consider a χ^2 test similar to that in [22]. In both cases, we apply the technique to the problem of HTTP attack detection. We also employ the pattern counting technique similar to that in [1]. Our goal is to provide a first line of defense by filtering out the vast majority of innocent HTTP traffic. Then potentially malicious traffic can be further analyzed with other sophisticated (and more costly) intrusion detection techniques.

The rest of the paper is organized as follows. In Chapter 2, we give an overview of network intrusion detection and discuss few examples of byte-level analysis techniques applied to web traffic. We also discuss Pearson’s χ^2 distance, which is the basis of one of our attack detection techniques. In Chapter 3, we cover the design of our detection techniques. Chapter 4 gives details about the datasets we use in our experiments to represent benign HTTP traffic and attack traffic. In Chapter 5 and 6, we present our results and evaluate the performance of our proposed techniques. In Chapter 7, we conclude the paper and discuss future work.

CHAPTER 2

Background

Development of methods for anomaly based intrusion detection is an important research problem in computer and network security. In this chapter, we discuss network intrusion detection systems and review some detection strategies applied to this problem. We also discuss the concept of Pearson's χ^2 statistic in general.

2.1 Network intrusion detection systems

A network intrusion detection system examines network traffic in real time and raises alarms when some potentially malicious traffic is detected [18]. Many existing intrusion detection systems are signature based [5, 19]. While such systems are able to detect known attacks effectively, they cannot prevent 'zero day' attacks or variants of known attacks.

An anomaly based detection strategy is capable of detecting unknown attacks. But such a technique is complex and, is more costly and tends to raise lot of false alarms. Designing an anomaly based detection strategy which is accurate and efficient, poses a significant challenge to security researchers.

2.2 Previous research

In this section, we discuss some intrusion detection techniques that employ byte level analysis of web traffic.

2.2.1 HMM-based detection

The paper [2] proposes a technique to model the structure of benign HTTP traffic using hidden Markov models [16]. Hidden Markov models or HMMs first appeared in [16] and were applied to problems in speech recognition. The training algorithm of an HMM models the structure of training data using a discrete hill climb approach [20]. The testing algorithm calculates probability of an input sequence with respect to the model to determine how well it fits the model [20].

The technique in [2] comprises of two phases—Training phase and Testing phase. In the training phase, n -gram sequences of bytes are extracted from benign HTTP traffic packets and are used to train a hidden Markov model. Some features of the training phase are mentioned below.

1. Multiple Classifiers – The authors propose the use of multiple HMMs instead of a single HMM to construct a more accurate model of benign HTTP payloads. Each HMM in the model has a different initial value for A , B and π matrices. Each HMM is trained on byte sequences extracted from the same training set of benign HTTP traffic. The training phase of a hidden Markov model is a discrete hill climb which stops at the local maxima. The authors claim to mitigate this problem with the use of multiple HMMs with different starting points, where some HMMs converge better than others. A final score for classifying an input sequence during testing phase, is a combination of scores given to the sequence by each HMM [2].
2. Number of hidden states – The number of hidden states of an HMM is equal to the size of n -grams used to train the model [2]. For example, if we want to score 5 grams of byte sequences, then the HMM should have 5 states.

The trained HMMs are expected to give a higher probability score to packets of benign HTTP traffic when compared to packets of attack traffic.

In the detection phase, an incoming HTTP packet is scored against trained HMMs. The packet is classified as benign or malicious based on the probability score computed by the HMMs. Figure 1 shows the detection phase of this technique.

Figure 1: HMMPayl IDS Architecture

N -gram sequences of bytes are extracted from payloads and each sequence is scored against an HMM. If the length of the payload is L bytes, then a total of $K = L - n + 1$ n -gram sequences can be extracted from it.

Figure 2 explains extraction of 5 grams from a packet of size 10 bytes. Six sequences of 5 grams are extracted from the packet and each sequence is scored against the HMM.

The probability score of the packet by the HMM is obtained by computing an arithmetic mean of scores of each n -gram sequence.

$$P(O|\lambda) = \frac{1}{K} \sum_{j=1}^K P(O_j|\lambda) \quad (1)$$

2	1	2	0	0	1	2	1	0	2	→ Sequence 1: 2-1-2-0-0
2	1	2	0	0	1	2	1	0	2	→ Sequence 2: 1-2-0-0-1
2	1	2	0	0	1	2	1	0	2	→ Sequence 3: 2-0-0-1-2
2	1	2	0	0	1	2	1	0	2	→ Sequence 4: 0-0-1-2-1
2	1	2	0	0	1	2	1	0	2	→ Sequence 5: 0-1-2-1-0
2	1	2	0	0	1	2	1	0	2	→ Sequence 6: 1-2-1-0-2

Figure 2: HMMPayl feature extraction scheme

In (1), O is the sequence of bytes extracted from a packet, K is the total number of n -gram sequences extracted from O and λ is a trained HMM.

The sequence O is scored against other HMMs using the same method described above. In Figure 3, x_i is the test packet, $s_{i,1...k}$ are scores of packet x_i computed by each of HMMs—HMM₁–HMM_k. The probability scores calculated by the HMMs are combined by computing an arithmetic mean , geometric mean , maximum or minimum of the scores. s_i^* is the final score of packet x_i computed by K HMMs in the model.

Figure 3: Packet scored against multiple HMMs

The final score of the packet is used to classify the packet as benign or malicious.

2.2.2 N -gram based detection

The paper [25] discusses an approach based on n -gram analysis of bytes in web traffic to detect attacks. They construct protocol specific models of benign traffic by analyzing frequency distributions of 1 gram of bytes from traffic captured on the following 6 ports: 20, 21, 22, 23, 25, 80.

All incoming traffic packets on a port are compared to the model of benign traffic for that port. A Mahalanobis distance [13] is calculated between frequency vector of incoming packet and mean of the model.

The authors of [25] present an improved version in [24] where they consider the use of site specific traffic to construct a model for benign traffic.

The paper [1] applies the n -gram based approach in [25] to the problem of file type classification where byte distributions of files are analyzed to classify an unknown file as a text, multimedia or an executable file. A model for executable files is built by extracting n -grams of bytes from training set of executable files. Mean and standard deviation are calculated for all unique n -grams of bytes that appear in the training set.

During the testing phase, an unknown file is processed to calculate relative frequency vector of n -grams of bytes. The relative frequency vector is compared to the model by computing its distance from mean vector of executable file model by using simplified Mahalanobis distance measure.

$$d(X, Y) = \sum_{i=1}^N \frac{|X_i - Y_i|}{\delta_i} \quad (2)$$

In (2), X is the frequency vector of the unknown file, Y is the mean vector, δ is

the standard deviation vector and N is the total number of n -grams.

As shown in Figure 4, if the Mahalanobis distance is larger than a certain threshold α , the file is classified as a multimedia file.

Figure 4: File type classification scheme

If the distance is smaller than α , then the file can be an executable file or a text file.

The authors propose another technique to separate text files and executable files. In this technique, a list of n -grams that appear in executable files is constructed from training set of executable files. The unknown file is processed to calculate the number of n -grams present in the list. The count is normalized by the file size. If the normalized count is greater than a threshold β , then the file classified as an executable file, else it is classified as a text file.

Such a technique works well because executable files have very large number of unique n -gram sequences compared to text files.

2.2.3 Other related work

The paper [15] considers the use of 2_v gram analysis to detect attacks in HTTP traffic. A 2_v gram is a group of 2 bytes which are v bytes apart in an HTTP packet. For a given distance of v , byte distributions of 2_v grams are used to train a one class SVM. If there are M different values of v , then M one class SVM models are trained on byte sequences from benign HTTP traffic.

In the detection phase, an unknown packet is processed and byte distributions are compared to M one class SVMs in the model. The final score given to the unknown packet is a combination of scores computed by each of the M one class SVMs.

The paper [12] discusses a χ^2 statistic based technique to detect malformed or malicious DNS traffic. They construct a model of benign DNS traffic by analyzing the type of DNS request, length of the request and byte distributions of the request. The byte distributions are analyzed using a χ^2 statistic test.

An incoming packet is processed to determine type of the request and length of the request. The byte distribution of the packet is compared to expected byte distribution for benign DNS traffic using a χ^2 test. A combined score based on request type, length and χ^2 test is computed for the unknown packet. This score is used to classify the packet as benign or malicious.

2.3 Pearson's χ^2 test

In this section, we review Pearson's χ^2 statistic which has been applied to the problem of HTTP attack detection in this paper.

Pearson's χ^2 test is a hypothesis test to determine closeness between 2 frequency distributions. The goal of such a test is to confirm whether an observed distribution

fits some expected distribution [6].

In hypothesis testing, one or more hypotheses are proposed initially. The initial hypothesis is referred to as null hypothesis and is denoted by H_0 . The alternative hypothesis is denoted by H_1 . The null hypothesis is assumed to be true at the beginning of the statistical test.

The result of the statistical test indicates whether the assumptions regarding the observed distribution are true or not. The hypotheses are accepted or rejected depending on the result of the statistical test. There are 2 types of errors associated with hypothesis testing

1. Type *I* error or False positive - An error of this type occurs when null hypothesis H_0 is falsely rejected when it should have been accepted. Type *I* error rate is denoted by α .
2. Type *II* error or False negative - An error of this type occurs when null hypothesis H_0 is falsely accepted when the alternative hypothesis H_1 should have been accepted. Type *II* error rate is denoted by β .

In this paper, we use a χ^2 statistic as an estimator function to determine difference between expected and observed distributions of n -grams of byte sequences in HTTP packets.

2.3.1 χ^2 Statistic

A χ^2 statistic between an observed distribution and an expected distribution is calculated as -

$$D^2(O, E) = \sum_{i=1}^N \frac{(O_i - E_i)^2}{E_i} \quad (3)$$

In (3), $D^2(O, E)$ is a χ^2 distance between an observed distribution O and an expected distribution E and $N-1$ is the degrees of freedom for the χ^2 test.

We apply the χ^2 statistic test on n -grams of byte sequences of HTTP packets. So, the number of degrees of freedom is equal to $(256^n - 1)$ for an n -gram size. In the detection phase, we classify an unknown packet as a benign packet or a malicious packet. A χ^2 distance is computed between observed n -gram distribution of the unknown packet and frequency distribution of benign HTTP packets.

The computed distance D^2 and degrees of freedom $N-1$ can be used to obtain a p-value from the χ^2 table. The p-value indicates the probability of the observed distribution to fit with the expected distribution. A large distance score indicates a lower p-value and a small distance score indicates a large p-value. Hence the distance scores themselves can be used directly instead of calculating p-value using a χ^2 table.

The computed distance D^2 is compared against a threshold— $\chi^2(\alpha, N - 1)$. $\chi^2(\alpha, N - 1)$ is a decision threshold for a χ^2 distribution with $N - 1$ degrees of freedom and a maximum type I error rate of α .

The null hypothesis H_0 and the alternative hypothesis H_1 are set up as

$$H_0 : D^2 > \chi^2(\alpha, N - 1)$$

$$H_1 : D^2 \leq \chi^2(\alpha, N - 1)$$

The unknown packet is assumed to be a malicious packet according to the null hypothesis H_0 . Let us assume that maximum false positive rate allowed by our technique is 0.05 ie No more than 5% of attack packets can be classified as benign

packets.

If D^2 score is greater than threshold $\chi^2(0.05, N - 1)$, then the null hypothesis is retained and the packet is classified as malicious. If the score is less than the threshold, then the null hypothesis is rejected and the packet is classified as benign.

CHAPTER 3

Design Overview

The goal of our project is to design and implement an accurate and efficient technique for detecting attacks in HTTP traffic. Our approach is based on byte level analysis of HTTP packets captured on the wire.

In the proposed solution, we build a model for benign HTTP traffic by analyzing frequencies of n -grams of bytes that appear in HTTP packets. In the detection phase, an incoming HTTP packet is compared to the model by computing a χ^2 statistic. The packet is classified as benign or malicious based on the χ^2 distance score. We also consider an adhoc n -gram distance measure and pattern counting described in [17] and [1], respectively, as alternative approaches to detect HTTP attacks.

3.1 χ^2 distance

In this technique, we compute a χ^2 statistic between observed n -gram frequency distribution of an unknown packet and expected frequency distribution of benign traffic. This approach comprises of 2 phases—Training phase and Testing phase. In the training phase, a model, representative of benign HTTP traffic, is constructed using frequency counts of n grams of bytes from benign HTTP packets.

1. Each HTTP packet in the training set is processed to obtain frequency counts of all unique n -grams of bytes.
2. The frequency count of each n -gram is normalized by the size of the packet to obtain a relative frequency count.

3. A frequency vector, using relative frequency counts of n -grams, is constructed for every packet.
4. A mean vector for the model is constructed by computing the arithmetic mean of relative frequencies for each n -gram across all packets in the training set.

The mean vector indicates the expected frequency distribution of bytes in a benign HTTP packet. Figure 5 shows the training and detection phase of χ^2 distance technique.

In the detection phase, an incoming packet is compared to the model of benign traffic to determine how much similar it is to benign HTTP traffic packets.

1. The incoming packet is processed to compute frequency of all n -grams of bytes in the packet.
2. The frequency count for an n -gram is normalized with the size of the packet to obtain a relative frequency count.
3. A frequency vector is constructed using relative frequency counts of all n -grams.
4. A χ^2 statistic is computed between frequency vector of the packet and the mean vector computed during the training phase.

$$D^2(X, Y) = \sum_{i=1}^N \frac{(X_i - Y_i)^2}{Y_i} \quad (4)$$

In (4), X is the frequency vector of the incoming packet, Y is the mean vector representative of benign traffic and N is the total number of n -grams being considered. If we consider 3 grams, then $N = (256)^3$. The degrees of freedom for χ^2 statistic is $N - 1$.

Figure 5: χ^2 Distance

If an n -gram appears in the incoming packet but does not appear in any of the packets in the training set, the Y_i component for that n -gram is 0 in (4). In this case, the denominator is smoothed by considering minimum value for an n -gram from the mean vector Y .

5. If the χ^2 distance score is lesser than a threshold α , then the packet is classified as a benign HTTP packet , else it is classified as a potentially malicious packet.

The χ^2 distance score of the packet determines the likelihood of the packet being a benign HTTP packet. Malicious packets are expected to have a larger distance from mean of distribution compared to benign HTTP packets.

The algorithm only needs to keep track of n -grams of bytes, which appear during training or testing phase.

3.2 Ad-hoc n -gram distance

This is an alternative distance measure we have used in this project. The distance between frequency vector and mean vector is calculated using (5).

$$d(X, Y) = \left(\sum_{i=1}^N |X_i - Y_i| \right)^2 \quad (5)$$

The training phase and testing phase of this approach are similar to the χ^2 distance based technique explained above.

3.3 Pattern Counting

This is an alternative technique we apply on the problem of HTTP attack detection. In this technique, we count the number of unique n -grams of bytes that appear in benign HTTP traffic . This approach is based on a technique to classify executable files and text files in [1].

1. Each packet in training set is processed to obtain unique n -grams of bytes.
2. In the end, we have a list of n -grams that appear in our training set of HTTP packets.
3. In the testing phase, we count occurrences of n -grams of bytes from an incoming packet which are present in the list of n -grams from training data.

4. The count of n -grams obtained in the previous step is normalized by packet size to obtain a relative count.
5. If the relative count is above a certain threshold value, then the packet is classified as a benign HTTP packet, else it is classified as a malicious packet.

Figure 6 shows training and detection phases of pattern counting technique. HTTP packets which are similar to those in training data are expected to have a higher relative count compared to other packets. A big advantage of such an approach is that it is extremely fast and simple compared to other n -gram based approaches, machine learning based techniques and other statistical techniques applied to the problem of intrusion detection.

Figure 6: Pattern Counting

CHAPTER 4

Datasets

We have used multiple datasets of benign HTTP traffic and attack traffic for the purposes of testing our techniques. In this chapter, we give details of the datasets and explain methods we have used to simulate realistic HTTP traffic.

4.1 Publicly available Datasets

In this section, we describe publicly available datasets of benign traffic and attack traffic which we have used for our experiments. Researchers often make their datasets publicly available for other researchers, students and scientific community to facilitate new research.

4.1.1 Benign Traffic

The authors of [2] provided us a dataset [8] of simulated HTTP traffic. The DARPA'99 [8] dataset consists of 5 days of benign HTTP traffic to and from an airforce base. The experiments of MIT's Lincoln Laboratory produced 3 weeks of network traffic. The aim of these experiments was to produce labeled datasets for network intrusion detection. The authors of paper [2] use HTTP traffic from the first week of DARPA dataset to evaluate their intrusion detection technique. We chose this dataset to construct models for benign HTTP traffic for our experiments. Table 1 gives details about the dataset.

Table 1: DARPA Dataset

Day	Number of TCP Packets	Number of Data Packets
1	161,602	28,187
2	196,605	34,446
3	189,362	33,051
4	268,250	44,185
5	150,847	26,315

4.1.2 Attack Traffic

We have used 3 datasets of HTTP attack traffic for our experiments. As with DARPA dataset, these datasets were used by the authors of paper [2].

4.1.2.1 Generic Attacks Dataset

This is a publicly available dataset of HTTP attacks provided by the authors of the paper [10]. It consists of 66 attacks which includes Denial of service attacks, URL decoding error attacks, shell code attacks and other attacks that cause information leakage. According to [2], the traffic in these datasets resembles benign HTTP traffic and hence these attacks are difficult to detect using statistical techniques.

4.1.2.2 Shellcode attacks Dataset

This dataset consists of 11 shellcode attacks. A shellcode attack tries to inject malware by exploiting a buffer overflow vulnerability on the target machine. These attacks are easier to detect as they contain byte sequences which are not present in most of benign HTTP packets. Benign traffic consists of ASCII printable characters and does not contain any executable bytes in the packets.

4.1.2.3 CLET attacks Dataset

CLET is a polymorphic worm generator which creates multiple encrypted versions of malware [7]. According to [2], CLET engine was used to create multiple versions of 8 shellcode attacks from Shellcode Attacks dataset.

Table 2 gives details about the attack datasets mentioned above.

Table 2: Attack Datasets

Name	Number of Attacks	Number of Data Packets
Generic Attacks	66	205
Shellcode Attacks	11	93
CLET Dataset	96	792

4.2 Simulated traffic datasets

We simulated large datasets of benign HTTP traffic and attack traffic for our experiments. We chose to simulate HTTP traffic rather than use real captured traffic to be sure there is no noise in the data used for our experiments.

4.2.1 Benign traffic

We simulated a dataset of benign HTTP traffic by setting up a wordpress [26] website. We created bots to generate HTTP requests towards the website and captured packets with tcpdump [23]. The traffic requests were generated using valid user-agents, valid HTTP methods and HTTP header parameters to make the simulated traffic as realistic as possible. Table 3 gives details about this dataset.

Table 3: Simulated Benign Traffic

Number of Packets
42,128

4.2.2 Attack Traffic

We simulated large datasets for 9 attacks to test our techniques. The challenge was to generate multiple versions of attacks that were realistic when compared to real attack traffic.

All attack traffic was generated using Metasploit [14]. Metasploit is a penetration testing framework which allows users to locate vulnerabilities in target systems and create bots to automate attacks. We used several in built modules of Metasploit to generate multiple versions of the same attack.

As our dataset of benign traffic was based on a wordpress [26] website, we felt that generating attacks related to php based vulnerabilities will be relevant for this project. We chose 9 attacks—4 php based attacks, 2 twiki based attacks, 3 shell code attacks—and created 100 versions for each attack. We divided the 9 attacks into three datasets. Here we give more details about each attack dataset and describe the techniques we used to make the traffic look as realistic as possible. Table 4 gives details about attack traffic datasets.

4.2.2.1 Php based Attacks Dataset

This dataset consists of 4 php based attacks. Wordpress is a php based technology. Attacks in this dataset are most relevant to corresponding dataset of simulated benign HTTP traffic.

1. WordPress plugin Foxypress Arbitrary Code Execution—A vulnerability in the foxypress plugin of wordpress software allows an attacker to upload a php script file on the server and execute code.
2. WordPress cache lastpostdate Arbitrary Code Execution—This vulnerability in wordpress software works when PHP 'register globals' option is enabled and allows an attacker to inject php code through a cookie parameter in an HTTP request.
3. PHP XML-RPC Arbitrary Code Execution—A flaw in the XML RPC module can be exploited by the attacker to embed OS commands in an xml file.
4. PHP Apache Request Headers Buffer Overflow—This vulnerability in PHP CGI allows an attacker to exploit a stack based buffer overflow from a parameter in the header of an HTTP request.

4.2.2.2 Twiki based Attacks

This dataset consists of 2 twiki based attacks.

1. Twiki search function arbitrary command execution—The search component of Twiki can be exploited to execute OS commands by passing them to the 'WebSearch' script.
2. Twiki History TwikiUsers Parameter Command Execution—The search component in Twiki can be used to inject commands by passing them to the 'TwikiUsers' script.

4.2.2.3 Microsoft based buffer overflow attacks

This dataset consists of 3 attacks which attempt to exploit buffer overflow vulnerabilities on Microsoft IIS server.

1. Microsoft IIS MDAC msadcs.dll buffer overflow—This vulnerability can be used to execute code when msadcs.dll and RDS factory service are exposed on IIS servers.
2. Microsoft IIS 5.0 IDQ buffer overflow -This vulnerability allows an attacker to exploit a stack based buffer overflow in IDQ ISAPI handler for Microsoft Index Server.
3. Microsoft IIS 5.0 Printer Host Header Overflow—This vulnerability exploits a buffer overflow in the request processor of the Internet Printing Protocol ISAPI module in IIS.

Table 4: Simulated Attack Traffic

Attack	Number of packets per attack
Php apache request headers buffer overflow	2
Wordpress Foxypress code execution	1,2
Wordpress lastpostdate code execution	2,3
Php XML-RPC code execution	1
Twiki Search function code execution	1
Twiki History function code execution	1
IIS MDAC msadcs.dll buffer overflow	1
IIS IDQ buffer overflow	3
IIS Printer Host buffer overflow	1

Figure 7 shows a benign HTTP request and the Figure 8 shows a request of a PHP Wordpress foxypress attack. We have constructed the HTTP header—Request URI , Host and User-Agent—in a way to make an attack resemble a benign HTTP request as much as possible.

```
GET /wordpress/ HTTP/1.1
User-Agent: Mozilla/5.0 (windows; U; windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824
Firefox/3.5.3 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: 192.168.122.93
Connection: Keep-Alive
Accept-Language: en-US,en;q=0.5
```

Figure 7: Benign HTTP Request

Metasploit provides in built modules to generate multiple variants of an attack. Some of the techniques we used to create variants of an attack are given below.

1. A different version of shell code is generated by metasploit for each attack attempt.
2. We limit the maximum size of TCP data segments for some attacks. For example, if the maximum tcp segment size is 400 bytes, then an attack of length 1100 bytes is divided across 3 packets of length 400 bytes , 400 bytes and 300 bytes. If there is no limit imposed on maximum tcp segment size, then the entire attack payload can be transmitted in 1 TCP packet.
3. We insert random bytes at the end of a URI of an attack request.

```
POST /wordpress/wp-content/plugins/foxypress/uploadify/uploadify.php.....HTTP/1.1
Host: 192.168.122.93
User-Agent: Mozilla /5.0 (x11; Ubuntu; Linux x86_64 rv:18.0) Gecko/20100101
Content-Type: multipart/form-data; boundary=_Part_255_2004455010_1755302731
Content-Length: 2924
```

```
--_Part_255_2004455010_1755302731
Content-Disposition: form-data; name="Filedata"; filename="5qQtor.php"
Content-Type: application/octet-stream

<?php ..$ipaddr='192.168.126.139';
..$port=16113;
..
...@set_time_limit(0); @ignore_user_abort(1); @ini_set('max_execution_time',0);
...$dis=@ini_get('disable_functions');
...if(!empty($dis)){
...$dis=preg_replace('/[ , ]+/', ' ', $dis);
...$dis=explode(',', $dis);
...$dis=array_map('trim', $dis);
...}else{
...$dis=array();
...}
...
...if(!function_exists('kDwaRuTBwkIFj')){
...function kDwaRuTBwkIFj($c){
....global $dis;
....
...if (FALSE !== strpos(strtolower(PHP_OS), 'win')) {
...$c=$c." 2>&1\n";
```

Figure 8: Sample Attack Request

CHAPTER 5

Experiments

We apply our techniques on datasets of benign HTTP traffic and attack traffic. Our aim is to filter out majority of benign HTTP traffic with simple and efficient techniques.

In [2], hidden Markov models were used to construct a model for benign HTTP traffic. The reported results indicate that their technique is slow and may not offer a practical solution to the problem. We perform experiments similar to the ones in [2] and evaluate our techniques on the basis of accuracy of detection and performance. We offer a direct comparison with results in [2] and show that our techniques give better results with substantial improvement in performance in terms of processing time.

5.1 Experimental Setup

The DARPA'99 dataset [8] was used to obtain packets of benign HTTP traffic. Raw bytes of HTTP traffic were extracted from TCP packets from tcpdump [23] packet capture (pcap) files.

A five fold cross validation, similar to the one in [2], was used to train the model for each technique. A five fold cross validation is used to obtain 5 times more results with limited amount of data available [11]. One day of traffic from DARPA dataset was used for training and traffic from remaining days was used for testing purposes. The experiment was repeated for each of the 5 days of traffic.

As cited in the previous chapter, datasets of attack traffic include Generic Attacks

dataset, Shellcode Attacks dataset and CLET dataset. These datasets are similar to the ones used in [2]. Each packet from the attack datasets was tested against a model of benign traffic constructed using the DARPA dataset.

We experimented with n -gram sizes 1–5. We observed that increasing the order of n -grams more than 5 did not give any substantial benefit in accuracy of detection and lead to increased processing times.

5.2 Performance Evaluation

The scores for benign traffic and attack traffic are plotted on a single graph for each attack dataset and n -gram size. The thresholds are varied based on the scores and number true positives and false positives are calculated for each threshold.

A true positive occurs when a benign packet is classified as benign packet by the technique. A false positive occurs when an attack packet is classified as a benign packet. For all our experiments, we compute a score for each packet and calculate a per packet detection rate.

True positive rate is plotted against false positive rate for each threshold. This graph is known as Receiver Operating Characteristics (ROC) curve [9]. ROC curves were used by electrical engineers in applications related to signal detection theory. An ROC curve measures the performance of a binary classifier by varying the threshold parameter.

The effectiveness of the test is determined by measuring area of the ROC curve also known as Area under the curve (AUC). An AUC of 1.0 indicates a perfect test with 0 false positives for any value of threshold. An AUC of 0.5 indicates linear increase of false positives with respect to true positives and indicates bad performance

of the classifier. AUC has been calculated using trapezoidal rule [3]. Trapezoidal rule approximates a region of a curve as a trapezoid and helps to calculate definite integral.

We calculate area under the ROC curve by not considering false positive rate greater than 0.1. The resulting area is called a partial area under the curve and is normalized by 0.1. Such a measure was used in [2] to evaluate results of their technique.

5.3 Experiments on Datasets of simulated HTTP traffic

We performed similar experiments on some additional datasets of HTTP traffic and attack traffic. As specified in chapter 4, all benign traffic was captured from a wordpress [26] website and attack traffic was generated using metasploit [14].

CHAPTER 6

Results

In this chapter, we present the results of our proposed techniques. We have used Receiver Operating Characteristic (ROC), Area Under Curve (AUC), and standard error to evaluate performance of the three techniques. For all the results presented here, the models were trained on benign HTTP traffic from DARPA'99 [8] dataset.

6.1 Generic Attacks Dataset

In this section, we discuss results of our techniques when applied on the Generic Attacks dataset. Figures 9, 11 and 13 show box plots of scores of benign traffic and attack traffic obtained by χ^2 distance, adhoc n -gram distance and pattern counting respectively. Figures 10, 12 and 14 show ROC curves obtained by plotting true positive rate and false positive rate for n -gram sizes—1–5 for each of the three techniques.

As cited earlier in chapter 4, attack traffic in this dataset resembles benign HTTP traffic. Hence the attacks are difficult to detect using statistical techniques.

Tables 5, 6, 7, summarize the results for all three techniques by providing the values of AUC, AUC_p and standard error. AUC_p denotes partial area under the curve obtained from ROC curves by considering false positive rates up to 0.1. The resulting area is normalized with 0.1 to obtain partial area under curve.

The AUC_p results from Tables 5 and 6 indicate that most of benign HTTP traffic is easily detected by our techniques using χ^2 distance measure and adhoc n gram distance measure. The adhoc n -gram distance performs much better compared to χ^2 distance for n -gram sizes 4 and 5.

Figure 9: BOX plot - Generic Attacks - χ^2 Distance- 3 grams

Figure 10: ROC - Generic Attacks - χ^2 Distance

Table 5: χ^2 distance - Generic Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9965	0.9778	0.00012
2	0.9977	0.9807	0.00008
3	0.9987	0.9897	0.00005
4	0.9958	0.9609	0.00014
5	0.9930	0.9346	0.00020

Figure 11: BOX plot - Generic Attacks- Adhoc n -gram Distance - 3 grams

Table 7 shows results for pattern counting approach. Results are very impressive for n -gram sizes 4 and 5 considering how simple and efficient the technique is. Results show that count of different 4 grams and 5 grams in a packet is enough to separate benign traffic and attack traffic.

Results are poor for n -gram sizes 1 and 2. But this is expected as all 1 grams and 2 grams of bytes from attack traffic packets are likely to be present in benign

Figure 12: ROC - Generic Attacks - Adhoc n -gram distance

Table 6: Adhoc n -gram distance - Generic Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9984	0.9887	0.000059
2	0.9985	0.9907	0.000057
3	0.9981	0.9903	0.000070
4	0.9980	0.9903	0.000072
5	0.9975	0.9899	0.000086

traffic as well.

Figure 15 shows the plot of AUC_p values for n -gram sizes 1–5 for all 3 techniques of our paper. Figure 16 shows a corresponding trend of AUC_p values from the paper [2]. Results indicate that our techniques perform much better for smaller n -gram sizes compared to the technique in [2]. Even a simple approach like pattern counting gives very good results.

Figure 13: BOX plot - Generic Attacks- Pattern Counting - 4 grams

Figure 14: ROC - Generic Attacks - Pattern Counting

Table 7: Pattern Counting - Generic Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.2760	0.0000	0.01000
2	0.7123	0.0002	0.00660
3	0.9822	0.8258	0.00055
4	0.9982	0.9843	0.00006
5	0.9993	0.9953	0.00002

According to paper [2], an n -gram-approach in [25] is outperformed by their HMM based technique. Our results indicate that n -gram techniques used in our paper give better results than the HMM based technique in [2].

Figure 15: Generic Attacks - AUC_p values for n -gram sizes 1–5

Figure 16: HMMPayl - Generic Attacks - AUC_p values for n -gram sizes 2–10

6.2 Shellcode Attacks Dataset

The shellcode attacks dataset contains attacks which inject executable code into the target machine. The attack traffic in this dataset contains many byte sequences which are not present in benign HTTP traffic. Most of the benign HTTP traffic used to train the model contains only ASCII characters in case of DARPA’99 [8] dataset. Attacks in this dataset are easier to detect compared to those in the Generic attacks dataset. Figures 17, 18, 19 show ROC curves obtained from all three techniques for n -gram sizes 1–5.

The results obtained from χ^2 distance are given in Table 8. The AUC_p values indicate that we obtain very good detection rate for all n gram sizes. The average AUC_p for n -gram sizes 1–5 is 0.99334 and standard deviation is 0.00383. An n -gram size of 1 byte is sufficient to detect most of benign HTTP traffic.

Figure 17: ROC - Shellcode Attacks - χ^2 distance

Table 8: χ^2 distance - Shellcode Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9995	0.9985	0.00002
2	0.9979	0.9928	0.00010
3	0.9978	0.9928	0.00010
4	0.9985	0.9947	0.00007
5	0.9978	0.9879	0.00010

Results for adhoc n -gram distance are shown in Table 9. The average AUC_p for n -gram sizes 1–5 is 0.99288 and standard deviation is 0.00008. These results are similar to those obtained by the χ^2 distance. Most of benign HTTP traffic is detected by small n -gram sizes.

Table 10 shows results obtained by applying the pattern counting. For this technique, 4 grams and 5 grams give very good results.

Figure 18: ROC - Shellcode Attacks - Adhoc n -gram distance

Table 9: Adhoc n -gram distance - Shellcode Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9979	0.9928	0.00010
2	0.9979	0.9928	0.00010
3	0.9978	0.9929	0.00010
4	0.9979	0.9930	0.00010
5	0.9979	0.9929	0.00010

Table 11 summarizes the results specified in paper [2] for this dataset. They conducted experiments using n -gram sizes 2–10. The average AUC_p values are shown for all rules used to combine scores computed by multiple HMMs. They also obtain very good results for this dataset. As mentioned earlier, most of benign HTTP traffic consists of only text data. Presence of executable byte sequences in attack traffic can be easily detected by any simple statistical technique.

Figure 19: ROC - Shellcode Attacks - Pattern Counting

Table 10: Pattern Counting - Shellcode Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.5634	0.0000	0.01268
2	0.9423	0.4423	0.00250
3	0.9660	0.6749	0.00150
4	0.9980	0.9928	0.00010
5	0.9982	0.9953	0.00009

Table 11: Results from HMMPayl - Shellcode Attacks

HMM combining Rule	Average AUC_p
Minimum	0.99909
Maximum	0.91937
Mean	0.97187
Geometric Mean	0.99967

An n -gram approach like ours is better for such a scenario compared to costly intrusion detection technique proposed in [2].

6.3 CLET Dataset

The CLET dataset contains polymorphic versions of malware from shellcode attacks dataset.

Figures 20 and 21 show ROC curves for χ^2 distance and adhoc n -gram distance. Table 12 summarizes results for χ^2 distance. The average AUC_p value for n -gram sizes 1–5 is 0.97266 and standard deviation is 0.03204. Table 13 shows results obtained from adhoc n -gram distance. The average AUC_p value for n -gram sizes 1–5 is 0.98812 and standard deviation is 0.01063.

AUC_p values indicate that both distance based techniques detect most of benign traffic easily. The adhoc n -gram distance outperforms χ^2 distance for n -gram sizes 4 and 5.

Table 12: χ^2 distance - CLET Dataset

N gram size	AUC	AUC_p	Standard Error
1	0.9982	0.9839	0.00004
2	0.9997	0.9976	0.00001
3	0.9997	0.9974	0.00001
4	0.9964	0.9633	0.00007
5	0.9922	0.9211	0.00014

Figure 22 shows ROC curves obtained from pattern counting for n -gram sizes 1–5. Table 14 shows the results for pattern counting when applied on this dataset. For n -gram sizes 3, 4, 5 all attacks are detected with 0 false positive rate.

Results from paper [2] are shown in Table 15. They conducted experiments using

Figure 20: ROC - CLET Dataset - χ^2 distance

Figure 21: ROC - CLET Dataset - Adhoc n -gram distance

Table 13: Adhoc n gram distance - CLET Dataset

N gram size	AUC	AUC_p	Standard Error
1	0.9968	0.9691	0.00007
2	0.9994	0.9928	0.00002
3	0.9991	0.9928	0.00002
4	0.9994	0.9930	0.00002
5	0.9990	0.9929	0.00003

Figure 22: ROC - CLET Dataset - Pattern Counting

n -gram sizes 2–10. The average AUC_p values are shown for all rules used to combine scores computed by multiple HMMs.

Table 14: Pattern Counting - CLET Dataset

N gram size	AUC	AUC_p	Standard Error
1	0.9721	0.7221	0.00045
2	0.9418	0.7133	0.00002
3	1	1	0
4	1	1	0
5	1	1	0

Table 15: Results from HMMPay1 - CLET Dataset

HMM combining Rule	Average AUC_p
Minimum	0.9984
Maximum	0.9355
Mean	0.97553
Geometric Mean	0.99935

6.4 Performance analysis

In this section, we discuss the performance of the three techniques in terms of packet processing time. Figure 23 shows the plot of per packet processing time against n -gram sizes for all three techniques considered in this paper. For χ^2 distance and adhoc n -gram distance, there is sharp increase in processing time for n -gram sizes greater than 3. Table 16 summarizes per packet processing time for each technique and n gram sizes 1–5.

Table 16: Summary of performance data (Time in milliseconds)

Technique	n -gram size				
	1	2	3	4	5
χ^2 distance	0.11068	0.65992	14.9650	18.0545	37.8059
Adhoc n -gram distance	0.11730	0.65510	6.5731	18.1456	39.2168
Pattern counting	0.03300	0.05660	0.0632	0.0664	0.0807

Figure 23: Performance - per packet processing time (milliseconds)

The performance of the pattern counting is the best. This is expected as the technique does not involve any processing at all. We have implemented pattern counting using hash tables. Table 17 shows number of unique n -grams from traffic of day one of DARPA'99 [8] dataset. The number of unique n -grams increases with n -gram size. But the increase is not large enough to cause collisions in our implementation of hash table. Hence, there is only a slight increase in processing time per packet with increase in n -gram size.

Table 18 shows the per packet processing time for the HMM-based technique in [2]. In some experiments, they randomly select a subset of n -grams from a packet and use them to calculate the probability score for the packet. This is done to improve performance at the cost of accuracy of detection. Sampling ratio is the percentage of n -grams selected from a packet.

Table 17: Number of unique n grams

N gram size	Count
1	90
2	3735
3	23756
4	58234
5	87495

Table 18: Summary of performance data - HMMPayl (Time in milliseconds)

Sampling Ratio	Time (ms)
20%	7.48
40%	12.71
60%	17.70
80%	22.96
100%	27.51

Results show that our techniques are faster for smaller n -gram sizes for χ^2 and adhoc n -gram distance. For our experiments, we consider all n -grams of an incoming packet to compute its similarity to benign traffic model.

We haven't provided any quantitative comparison between both approaches in terms of memory requirements. But the approach in [2] would require a lot more memory to maintain multiple HMMs with large number of states, compared to our approach which just requires frequency counts of n -gram byte sequences.

Details about the system used for our testing purposes are given in Table 19.

Table 19: System Configuration

CPU	Memory	Operating System
Intel Core i-5 2520M	4 GB	Microsoft Windows 7.0

6.5 Results for experiments on dataset of simulated HTTP traffic

In this section, we present results from experiments on datasets of simulated benign HTTP traffic and attack traffic. As mentioned in chapter 4, benign HTTP traffic was captured from a wordpress website we had set up.

We used 4 attack datasets—Php based attack dataset, Twiki based attack dataset, Microsoft IIS Buffer Overflow Attacks dataset and Generic attacks dataset. The Generic attacks dataset is same as the one used in experiments with DARPA’99 [8] dataset. More details about these datasets are given in chapter 4.

6.5.1 Experimental Setup

A five fold cross validation [11] was implemented on the dataset of benign HTTP traffic where 80% of traffic was used for training purposes and remaining 20% was used for testing purposes. The experiment was repeated five times. Each fold of test data was compared against an attack dataset.

6.5.2 Results

In this section, we summarize results by providing ROC curves and values of AUC.

6.5.2.1 Generic Attacks Dataset

Figure 24 shows ROC curves derived by using the χ^2 distance. The corresponding AUC values are shown in Table 20. The detection system is able to detect all attacks with no false positives for n -gram sizes 3,4 and 5.

Figure 24: ROC - Generic Attacks - χ^2 Distance

Table 20: χ^2 distance - Generic Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9976	0.9825	0.00017
2	0.9999	0.9997	0.00001
3	1	1	0
4	1	1	0
5	1	1	0

Figure 25 shows ROC curves derived using adhoc n -gram distance. The AUC values in Table 21 indicate that the adhoc n -gram distance does not perform better than the χ^2 distance based approach.

Figure 25: ROC - Generic Attacks - Adhoc n -gram Distance

Table 21: Adhoc n -gram distance - Generic Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9981	0.9863	0.00016
2	0.9989	0.9933	0.00011
3	0.9989	0.9934	0.00011
4	0.9990	0.9942	0.00011
5	0.9991	0.9949	0.00010

Figure 26 shows ROC curves for pattern counting for n -gram sizes 1–5. The corresponding AUC values are given in Table 22. We obtain very good results for n -gram sizes 3,4 and 5.

Figure 26: ROC - Generic Attacks - Pattern Counting

Table 22: Pattern Counting - Generic Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.4634	0	0.00930
2	0.9756	0.7564	0.00091
3	1	1	0
4	1	1	0
5	1	1	0

6.5.2.2 Php based Attacks Dataset

Attacks in this dataset are based on Php related vulnerabilities and are most relevant to the benign traffic used to train the model. Figure 27 shows ROC curves obtained by χ^2 distance for this dataset. Table 23 shows the values of AUC for different n -gram sizes. All attacks are detected with zero false positives for n -gram

sizes 2 or more using χ^2 distance.

Figure 27: ROC - Php based Attacks - χ^2 Distance

Table 23: χ^2 distance - Php based Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9998	0.9986	0.00004
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

Table 24 summarizes results obtained from adhoc n -gram distance derived from ROC curves in Figure 28. Results are equally good when compared to those obtained by χ^2 distance.

Figure 28: ROC - Php based Attacks - Adhoc n -gram Distance

Table 24: Adhoc n -gram distance - Php based Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9996	0.9944	0.00006
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

Pattern counting also works well on this dataset of attacks as evident from Table 25. The ROC curves corresponding to these results, are shown in Figure 29.

Figure 29: ROC - Php based Attacks - Pattern Counting

Table 25: Pattern Counting - Php based Attacks

N gram size	AUC	AUC_p	Standard Error
1	0.9872	0.8728	0.00043
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

6.5.2.3 Microsoft IIS buffer overflow Attacks Dataset

This dataset contains shellcode attacks which attempt to exploit buffer overflow vulnerabilities on Microsoft IIS servers. Attacks in this dataset are a lot easier to detect when compared to benign traffic. This is evident from ROC curves in Figures 30, 31 and 32. The corresponding values of AUC and AUC_p are given in Ta-

bles 26, 27 and 28. We obtain perfect detection for all n -gram sizes for all techniques.

Figure 30: ROC - MS Buffer overflow Attacks - χ^2 Distance

Table 26: χ^2 distance - MS Buffer overflow Attacks

N gram size	AUC	AUC_p	Standard Error
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

Figure 31: ROC - MS Buffer overflow Attacks - Adhoc n -gram Distance

Table 27: Adhoc n -gram distance - MS Buffer overflow Attacks

N gram size	AUC	AUC_p	Standard Error
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

Figure 32: ROC - MS Buffer overflow Attacks - Pattern Counting

Table 28: Pattern Counting - MS Buffer overflow Attacks

N gram size	AUC	AUC_p	Standard Error
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

6.5.2.4 Twiki based Attack Datasets

Results for this dataset are similar to those obtained from Microsoft Shellcode attacks dataset. The detection system is able to separate benign traffic from attack traffic for all n -gram sizes for all techniques. The ROC curves are given in Figures 33, 34 and 35. Tables 29, 30 and 31 summarize results by providing values of

AUC, AUC_p for n -gram sizes 1–5.

Figure 33: ROC - Twiki based attacks - χ^2 Distance

Table 29: χ^2 distance - Twiki based Attacks

N gram size	AUC	AUC_p	Standard Error
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

Figure 34: ROC - Twiki based attacks - Adhoc n -gram Distance

Table 30: Adhoc n -gram distance - Twiki based Attacks

N gram size	AUC	AUC_p	Standard Error
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

Figure 35: ROC - Twiki based attacks - Pattern Counting

Table 31: Pattern Counting - Twiki based Attacks

N gram size	AUC	AUC_p	Standard Error
1	1	1	0
2	1	1	0
3	1	1	0
4	1	1	0
5	1	1	0

CHAPTER 7

Conclusion and Future work

In this project, we implemented three n -gram based techniques to detect attacks in HTTP traffic. Our techniques are based on analysis of byte distributions in HTTP packets. The techniques were extensively tested on standard public datasets of HTTP traffic and attack traffic. Results indicate that our approaches can detect 99% of innocent HTTP traffic from attack traffic in case of DARPA'99 dataset [8].

We also compare our results with those found in [2] in terms of accuracy of attack detection and performance. Results clearly indicate that our techniques perform much better than their HMM-based approach for smaller n -gram sizes. Our techniques are also much simpler and efficient in terms of processing time and memory requirements.

Our method relies on a single model of benign HTTP traffic and compares all incoming traffic with this model. It might be useful to create multiple content specific models of HTTP traffic. All incoming packets can be compared with each model before classifying them as benign or malicious.

Our approach classifies each individual packet as benign or malicious. Incoming packets which belong to the same TCP stream can be re-assembled into a single stream of bytes. This stream of bytes can then be classified as benign or malicious. Such an approach can give more accurate detection.

LIST OF REFERENCES

- [1] I. Ahmed, K. Lhee, Classification of packet contents for malware detection, *Journal of Computer Virology*, 2011, Volume 4, Issue 7, pp. 279–295
- [2] D. Ariu, R. Tronci, and G. Giacinto, HMMPayl: An intrusion detection system based on hidden Markov models, *Computers and Security*, 2011, Volume 30, Issue 4, pp. 221–241
- [3] K. E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons , New York , 1989
- [4] J. Aycock, *Computer Viruses and Malware*, Springer, New York, 2006
- [5] Bro, Bro Intrusion Detection System
www.bro.org
- [6] H. Chernoff, E. L. Lehmann. The Use of Maximum Likelihood Estimates in χ^2 Tests for Goodness of Fit, *The Annals of Mathematical Statistics*, 1954, Volume 25, Issue 3, pp. 579–586
- [7] CLET , CLET Polymorphic shellcode engine
<http://www.phrack.org/issues.html?issue=61&id=9>
- [8] DARPA'99 dataset for intrusion detection
<http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1999data.html>
- [9] T. Fawcett, ROC graphs: An introduction to ROC analysis, *Pattern Recognition Letters*, 2006 , Volume 27 , Issue 8 , pp. 861–874
- [10] K. Ingham, H. Inoue, Comparing anomaly detection techniques for HTTP, *Recent Advances in Intrusion Detection*, 2007, Volume 4637, pp. 42–62
- [11] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995, Volume 2, pp. 1137–1143
- [12] C. Kruegel, T. Toth, E. Kirda, Service specific anomaly detection for network intrusion detection, *ACM Symposium on Applied Computing, Madrid*, 2010, pp. 201–208
- [13] P. C. Mahalanobis, On the generalised distance in statistics, *Proceedings of the National Institute of Sciences of India*, 1936, Volume 2, pp. 49–55

- [14] Metasploit, Metasploit Penetration Testing Tool
<http://www.metasploit.com/>
- [15] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, W. Lee. McPAD : A Multiple Classifier System for Accurate Payload-based Anomaly Detection, *Computer Networks—Special Issue on Traffic Classification and Its Applications to Modern Networks*, 2009, Volume 6, Issue 5, pp. 864–881
- [16] L. Rabiner (1989). A tutorial on hidden Markov models and selected applications in speech recognition, 1989, *Proceedings of the IEEE* , Volume 77, Issue 2, pp. 257–286
- [17] N. Runwal, R. Low, M. Stamp, Opcode graph similarity and metamorphic detection, *Journal of Computer Virology*, 2012 , Volume 8, Issue 1–2, pp. 37–52
- [18] K. Scarfone, P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS), *Computer Security Resource Center, NIST* , 2007, 800–94
- [19] Snort, Snort Open source Network Intrusion Detection System
<http://www.snort.org>
- [20] M. Stamp, A revealing introduction to hidden Markov models,
<http://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf>
- [21] Symantec Internet Security Threat Report, Volume 17
<http://www.symantec.com/threatreport/>
- [22] A. Toderici, M. Stamp, Chi-squared distance and metamorphic virus detection, *Journal of Computer Virology and Hacking Techniques*, 2013 , Volume 9, Issue 1, pp. 1–14
- [23] TCP Dump - Packet Capture Library
<http://www.tcpdump.org>
- [24] K. Wang, G. Cretu, S. Stolfo, Anomalous payload based worm detection and signature generation, *Recent Advances in Intrusion Detection*, 2005, Volume 3858 , pp. 227–246
- [25] K. Wang, S. Stolfo, Anomalous Payload-Based Network Intrusion Detection, *Recent Advances in Intrusion Detection*, 2004, Volume 3224, pp. 203–222
- [26] Wordpress, Wordpress blogging software
www.wordpress.org